Sub-behaviour relations for session-based client/server systems

Sub-behaviour relations for session-based client/server systems

Franco Barbanera Ugo de'Liguoro

April 5, 2013

Abstract

We propose a refinement and a simplification of the behavioural semantics of session types, based on the concepts of compliance and sub-behaviour from the theory of web contracts. We introduce three relations on a suitable class of behaviours with higher-order input/output, called "session behaviors". Such relations, depending on each other, represent the idea of sub-behaviour from the point of view of a client, a server or a peer, respectively. A restriction of the intersection of the first two characterizes the usual sub-behaviour relation (from the literature). We then device a formal system for three subtyping relations (dubbed CSP-subtyping) for session types that takes into account the role played by a user of a channel during an interaction, so extending Gay and Hole subtyping theory. We show that our session behaviors and sub-behaviour relations provide a sound and complete semantics for CSP-subtyping (and for Gay and Hole subtyping as a by-product).

1 Introduction

A great deal of work is presently devoted to the formalisation of interaction through the network, widening the research area on protocols and investigating its basic concepts. This is motivated by the impressive growth of Web based systems and by the development of service-oriented programming. In this scenario programmers are expected to produce modules that heavily depend on the communication with systems and programs written by third parties, of which nothing is known except a generic and often informal description of their behaviour.

Session types and contracts are two formalisms used to study client/server protocols. Session types have been introduced in [17] as a tool for statically checking safe message exchanges thorough channels. The basic concept is that of *session*, which is a logic unit collecting and structuring messages exchanged among a set of agents, sharing private channels to prevent interference by unchecked third parties. Session types represent the usage of each session channel by a regular tree of types (which is itself considered as a type), abstractly representing all sequences of actions of which the typed channel is the "subject" in the π-calculus jargon. In the theory of session types each type has a dual, describing the same interaction from the point of view of the process holding

the opposite end of the channel; the exact correspondence of dual typings of the same channel ensures error freeness (but not deadlock freeness: see [14]).

Contract theory, as proposed in [7, 18, 11], addresses the problem of abstractly describing behavioural properties of systems by means of process algebra. In the case of first order theory, namely without input/output of non atomic data, contracts are formalised by means of a subset of CCS without τ terms from [13]. These include, beside prefixing and recursion, external and internal choice, denoted by $\sigma + \sigma'$ and $\sigma \oplus \sigma'$ respectively, but not parallel composition, which has a computational meaning. Indeed a contract is a static object, that is some kind of abstract interface offered by a server to its possible clients, describing the server overall behaviour during an interaction. To formally check whether a client will comply with the server, a dual contract can be used, describing this time the client requests. This leads to an asymmetric view of the client/server interaction, because of a bias to the client side: it is only the client that is expected to complete in any interaction with the server, and not vice versa.

A benchmark for comparing the two approaches is flexibility. If a service is looked at in a third party's server it is unlikely that anything will be found that perfectly matches the query. Hence criteria are needed to decide whether what is available from the server can be safely used on the client side, without expecting the exact matching of descriptions in the respective interfaces. Moreover, such criteria should be checked automatically, since the request of a service might well occur at run time.

On the session type side a natural solution is polymorphism and more precisely subtyping, firstly proposed in [16]. This is an extension of input/output subtyping from [22], such that e.g. the branching type $\&\langle \ell_1 : A \rangle$ is considered as a subtype of the larger $\&\langle \ell_1 : A, \ell_2 : B \rangle$ because anything offering the choice between messages or data of type A and B via the options ℓ_1, ℓ_2 can safely masquerade, either as a server or as a client, offering the option ℓ_1 only. Dually, the selection type $\oplus\langle \ell_1 : A, \ell_2 : B \rangle$ signals the possibility of choosing among ℓ_1 and ℓ_2 without committing to either of them; then this is a subtype of $\oplus\langle \ell_1 : A \rangle$ because any agent satisfying the stronger constrain of choosing ℓ_1 will safely do in any environment correctly reacting to both ℓ_1 and ℓ_2.

A concept of sub-contract can be defined by adapting the theory of testing from [12]. In [18, 20] a *compliance* relation $\rho \dashv \sigma$ is introduced which ensures that any request from the client ρ is satisfied by the server σ, so that any possible interaction among ρ and σ will never prevent the client from completing; this can be seen as the success condition of testing σ against ρ. Compliance naturally induces a preorder, $\sigma_1 \preceq_s \sigma_2$ in our notation, which correspond to the inclusion $\mathsf{Client}(\sigma_1) \subseteq \mathsf{Client}(\sigma_2)$, where $\mathsf{Client}(\sigma) = \{\rho \mid \rho \dashv \sigma\}$. We call $\preceq_s$ the *server sub-behaviour* relation, which essentially coincides with the sub-contract relation in [18].

In [3] we pointed out that a dual notion can be considered: $\rho_1 \preceq_c \rho_2$, corresponding to the inclusion $\mathsf{Server}(\rho_1) \subseteq \mathsf{Server}(\rho_2)$, where $\mathsf{Server}(\rho) = \{\sigma \mid \rho \dashv \sigma\}$. We call $\preceq_c$ the *client sub-behaviour* relation.

Actually a third notion can be taken into account by considering interactions where components have no commitments to each other. When no asimmetry is imposed on interacting components they can then be looked at as *peers*. An orthogonality relation $\perp$ formalizing this sort of interaction could hence be defined as $\dashv \cap \vdash$, a relation strongly related to the subsieve relation in [8]. Then

$\sigma \perp \sigma'$ holds if both σ and σ' complete in any terminating interaction among the two and induces a preorder $\preceq_*$ corresponding to the inclusion $\mathsf{Peer}(\sigma_1) \subseteq \mathsf{Peer}(\sigma_2)$, where $\mathsf{Peer}(\sigma) = \{\tau \mid \tau \perp \sigma\}$.

The notion of compliance was also investigated in [10] where, however, the asymmetric nature of compliance has been put aside, by considering only a relation similar to $\perp$.

The relations $\preceq_c$, $\preceq_s$ and $\preceq_*$ hence represent the notion of substitutability between, respectively, clients, servers and peers in client/server-based distributed systems taking into account the different roles a component plays in an interaction. Intuitively, if we look at σ_1 and σ_2 as the behaviours of two servers, then, in case $\sigma_1 \preceq_s \sigma_2$ holds, σ_2 represents the behaviour of a server offering *richer* services (which means also possibly longer interactions) than the server described by σ_1. In case, instead, we look at two behaviours ρ_1 and ρ_2 as the descriptions of two clients, the relation $\rho_1 \preceq_c \rho_2$ states that the client represented by ρ_1 is *less demanding* (in the sense also of the number of consecutive requests) than the client with behaviour ρ_2. Whenever $\sigma_1 \preceq_* \sigma_2$, instead, a component with behaviour σ_2 can safely interact in a symmetric manner with at least all the peers of σ_1.

For what concerns the operators $+$ and $\oplus$, the $\preceq_*$ relation behaves the same as $\preceq_p$ for $p = c$, namely covariantly in the number of $+$ summands and contravariantly in the number of the $\oplus$-summands (and covariantly w.r.t. the respective continuations), that is session behaviors and behavioural subtyping mirror branching and selection session types w.r.t. (syntactic) subtyping, respectively. To these properties, however, the sub-behaviour relations $\preceq_c$ and $\preceq_s$ add dual forms of subtyping *in depth* that does not hold in the subtyping theory for session types deviced by Gay and Hole in [16].

In a first-order setting, i.e. where components themselves cannot be freely exchanged in a system, the three sub-behaviour relations can be formalized independently. When higher-order is considered, however, they become intimately correlated, since also the exchanged components can play a particular role in the system, not necessarily corresponding to the role of the components that exchanged them, as we shall see later on in an example.

Turning back to the comparison of session types with contracts, there is a natural interpretation of the first order session type A into a contract $[\![A]\!]$, which interprets branching types $\&\langle\cdots\rangle$ into external choices, and selection types $\oplus\langle\cdots\rangle$ into internal choices. However, as it has been observed in [19], such an interpretation is unsound. Indeed $[\![\&\langle\ell_1 : \mathsf{end}\rangle]\!] = \ell_1.\mathbf{1}$ and $[\![\&\langle\ell_1 : \mathsf{end}, \ell_2 : A\rangle]\!] = \ell_1.\mathbf{1} + \ell_2.\tau$, where $\mathbf{1}$ is the contract of completed processes and $\tau = [\![A]\!]$. Now, in the subtyping theory we have $\&\langle\ell_1 : \mathsf{end}\rangle \leq \&\langle\ell_1 : \mathsf{end}, \ell_2 : A\rangle$, but $\ell_1.\mathbf{1} \not\preceq_* \ell_1.\mathbf{1} + \ell_2.\tau$ (and this is the case for $\preceq_c$ and $\preceq_s$ as well). In fact, if we take $\rho = \overline{\ell_1}.\mathbf{1} + \overline{\ell_2}.\sigma$, for any σ unrelated to τ, then $\rho \in \mathsf{Peer}(\ell_1.\mathbf{1}) \setminus \mathsf{Peer}(\ell_1.\mathbf{1} + \ell_2.\tau)$. Problems arise also because of the admittance of unguarded recursion, e.g. $\mathsf{rec}\, x.x$, in the contract syntax.

The conclusion we draw from the above remarks is that contracts are a larger and likely more expressive formalism than session types. This does not rule out the possibility of using contracts to give meaning to session types, which is what we shall begin the technical part of the present paper with: on the contract side we shall define a formal language of *session behaviors* and denote it by $\mathcal{SB}$.

It can be looked at as the image of the session types interpretation map, so that in $\sum_i a_i.\sigma_i$ and in $\bigoplus_i \overline{a}_i.\sigma_i$ the a_i are pairwise distinct and names a_i and co-names $\overline{a}_i$ occur within sums $+$ and $\oplus$ only, respectively. Besides, recursion is assumed to be contractive. By doing so we trim the nondeterminism of the system so that the previous example ceases to be problematic. In fact now $\ell_1.\mathbf{1} \preceq_* \ell_1.\mathbf{1} + \ell_2.\tau$ holds, since $\rho = \overline{\ell_1}.\mathbf{1} + \overline{\ell_2}.\sigma$ is ruled out from the set of possible clients (and hence of peers) of $\ell_1.\mathbf{1}$, being syntactically incorrect. This implies for instance: $\ell_1.\mathbf{1} \preceq_s \ell_1.\ell_3\mathbf{1} + \ell_2.\tau$. The syntactical restriction we impose on session behaviours is tantamount to restrict the nondeterminism in their interactions, so that it depends at any time on just one of the two actors of an interaction.

Although session behaviours are isomorphic to session types, we keep them distinct from types, because the former have an operational semantics. We shall describe such an operational semantics by means of an LTS defining a reduction relation among parallel composition of contracts, such that $\rho \| \sigma \Longrightarrow \rho' \| \sigma'$ if $\rho \stackrel{\alpha}{\Longrightarrow} \rho'$ and $\sigma \stackrel{\tilde{\alpha}}{\Longrightarrow} \sigma'$, and α and $\tilde{\alpha}$ are actions that synchronise with each other. The compliance relation $\rho \dashv \sigma$ is then defined by the clause that, if $\rho \| \sigma \Longrightarrow \rho' \| \sigma' \not\Longrightarrow$, then $\rho' = \mathbf{1}$, formally describing the fact that the "server" σ has satisfied all the requests of the client ρ. Notice that, according to our client/server/peer roles, σ does not need to "complete" (i.e. to reduce to $\mathbf{1}$) unless also $\rho \vdash \sigma$ holds; besides, we shall admit also non terminating complying interactions.

We provide now an intuition of the intended meaning of the notions described above by modifying an example sketched in [3] which, in turn, is an adaptation of one in [19]. Let us consider a *Ballot-Service*. This service can receive a login and, if correct, signal to the client (a voter), by means of the message `Ok`, that it is enabled to vote for one of the candidates A, B or C. After that, the server offers also the possibility of voting for one of two possible vice-candidates. In case the login is incorrect, instead, a message `Wrong` is issued to the client. By means of recursion, a voter is allowed to retry the login action in case of a failure. The following element of $\mathcal{SB}$ then abstractly describes the behaviour of the *Ballot-Service*:

$$\begin{aligned}\mathsf{BallotServiceBeh} = \mathsf{rec}\, x.\, \mathtt{Login}.(\overline{\mathtt{Wrong}}.x \oplus\ & \overline{\mathtt{Ok}}.(\mathtt{VoteA}.(\mathtt{Va1} + \mathtt{Va2}) \\ & + \\ & \mathtt{VoteB}.(\mathtt{Vb1} + \mathtt{Vb2})) \\ & + \\ & \mathtt{VoteC}.(\mathtt{Vc1} + \mathtt{Vc2}))\ )\end{aligned}$$

The following element of $\mathcal{SB}$, instead, describes the behaviour of *Voter1*, a possible client of our *Ballot-Service*:

$$\mathsf{Voter1Beh} = \overline{\mathtt{Login}}.(\mathtt{Wrong} + \mathtt{Ok}.(\overline{\mathtt{VoteA}} \oplus \overline{\mathtt{VoteB}}))$$

As said before, the notion of compliance expresses the idea that a client is entitled to abandon the interaction at any time, while a server is expected to react properly to all client requests. So we have that

$$\mathsf{Voter1Beh} \dashv \mathsf{BallotServiceBeh}$$

Notice that *Voter1* is not prepared to vote for vice-candidates and it is prepared to try and login just once, notwithstanding the server admits repeated login actions after possible failures.

Let us now take into account a different Ballot Service, *Ballot-Service-2*, whose abstract behaviour is described by the following element of $\mathcal{SB}$:

$$\mathsf{BallotService2Beh} = \mathtt{Login}.(\overline{\mathtt{Wrong}} \oplus \ \overline{\mathtt{Ok}}.(\mathtt{VoteA} + \mathtt{VoteB}))$$

A voter that uses *Ballot-Service-2* can just try once to login and can vote for just A or B, without the possibility of choosing any vice-candidate.
Intuitively, *Ballot-Service* offers then a "richer" service that *Ballot-Service-2*, and in any system designed to have *Ballot-Service-2* as ballot service, we can safely replace it by *Ballot-Service*. This safe-sostitutability property is guaranteed by the fact that, according to our sub-behaviour relations, we have

$$\mathsf{BallotService2Beh} \preceq_s \mathsf{BallotServiceBeh}$$

In fact any voter complying with a ballot service allowing for just one login, two candidates choices and no vice-candidates, will definitely comply with the more liberal one that allows for more login attempts, an extra choice of candidates and also the possibility of choosing a vice-candidate after the candidate choice.

In order to exemplify the $\preceq_c$ relation, let us now consider another possible voter, *Voter2*, who wishes to return a blank ballot, i.e. to participate to the ballot but without voting for any candidate. The behaviour of *Voter2* is described by the following element of $\mathcal{SB}$:

$$\mathsf{Voter2Beh} = \overline{\mathtt{Login}}.(\mathtt{Wrong} + \mathtt{Ok})$$

According to the $\preceq_c$ relation we have:

$$\mathsf{Voter1Beh} \preceq_c \mathsf{Voter2Beh}$$

because a voter described by Voter2Beh is less demanding than one described by Voter1Beh, that is a voter behaving as Voter2Beh complies with all the ballot servers voters described by Voter1Beh comply with. This implies that any voting system designed to accommodate voters with the behaviour Voter1Beh can safely accommodate voters having the behaviour Voter2Beh. An example of use of the relation $\preceq_*$ will be provided later on.

If, as in the examples above, only first-order session-behaviours were taken into account, the relations $\preceq_c$, $\preceq_s$ and $\preceq_*$ would not be difficult to formalize and investigate. As we shall see, difficulties arise, instead, when we wish to use our formalism to describe the behaviour of components that have the possibility of exchanging among themselves other components (with their corresponding behaviours), i.e. when we consider *higher-order* session behaviours. In the context of session types, the exchange of components corresponds to delegation, realized by means of channel passing, and it has been extensively investigated since the very beginning of the development of session types theory.

In the present paper we shall define the set $\mathcal{SB}$ of session behaviours in order to take into account also higher-order behaviours, that is behaviours with higher-order input actions, $?(\sigma^p)\sigma'$, and higher-order output actions $![\sigma^p]\sigma'$, where p is one of the three possible role played by the sent/received behaviour σ: c (client), s (server) or $*$ (peer). Whereas in the context of session types we can look at σ as the semantics of the type of a sent/received channel, from a more general point of view we can look at it as the description of a sent/received component.

The use of polarities as superscripts of received and sent contracts in $\mathcal{SB}$ is justified by the asymmetry of the approach considered here: without marking received and sent contracts with polarities, incorrect situations easily arise (see also [2]). For instance, let us assume that a communicating agent is waiting for a component (or a channel) enabling an interaction as a client according to the behaviour σ. If it actually receives a component conforming to τ and τ describes a *more demanding* client behaviour, the received component could make requests that the server to which the agent is connected (or will be connected to) is not able to satisfy. So the information that τ and σ have to be confronted w.r.t. the relation $\preceq_c$ is essential, and it is provided by the label c.

Notice that the only possibility of avoiding the use of labels c or s in sent and received behaviours is to force their contracts to coincide, or to be related by $\preceq_*$. This, however, would severely restrict the flexibility of our formalism.

Then, when we formally define the compliance relation $\dashv$, we have to take into account that the interaction with a higher-order session behaviour $![\sigma_1^c]\sigma_2$ can be triggered by any $?(\tau_1^c)\tau_2$ such that $\tau_1 \preceq_c \sigma_1$ and not just when $\tau_1 \equiv \sigma_1$ or $\tau_1 \preceq_* \sigma_1$. A similar motivation similar justifies the fact that $![\sigma_1^s]\sigma_2$ can be triggered by any $?(\tau_1^s)\tau_2$ such that $\tau_1 \preceq_s \sigma_1$. In our formalism we shall further ask received/sent behaviours not to contain free variables, for reasons that will be clarified in the paper.

For the time being, let us consider a system where the *Ballot-Service* is modularized into an *Authentication-Service* and a *Vote-Bookkeeper* service. The *Authentication-Service* checks the right of the client to vote, and then passes to her the description of how her vote can be expressed (this can be a channel through which the vote can be given, or a component enabling the voter to transparently interact, as a client, with the *Vote-Bookkeeper*.) The behaviour of the *Authentication-Service* is described by the following higher-order element of $\mathcal{SB}$:

$$\begin{array}{l}\mathsf{AuthServiceBeh} = \\ \quad \mathtt{rec}\ x.\ \mathtt{Login}.\big(\ \overline{\mathtt{Wrong}}.\ x \\ \qquad\qquad \overset{\oplus}{} \\ \qquad\qquad \overline{\mathtt{Ok}}.\ \mathbf{!}[\ (\ \overline{\mathtt{VoteA}}.(\overline{\mathtt{Va1}}\ \overset{\oplus}{}\ \overline{\mathtt{Va2}}) \oplus \overline{\mathtt{VoteB}}.(\overline{\mathtt{Vb1}}\ \overset{\oplus}{}\ \overline{\mathtt{Vb2}}) \oplus \overline{\mathtt{VoteC}}.(\overline{\mathtt{Vc1}}\ \overset{\oplus}{}\ \overline{\mathtt{Vc2}})\ )^{\boldsymbol{c}}\ ]\ \big)\end{array}$$

whereas the *Vote-Bookkeeper* behaves according to:

$$\mathsf{VoteBookprBeh} = \mathtt{VoteA}.(\mathtt{Va1} + \mathtt{Va2}) + \mathtt{VoteB}.(\mathtt{Vb1} + \mathtt{Vb2}) + \mathtt{VoteC}.(\mathtt{Vc1} + \mathtt{Vc2})$$

As said before, the use of polarities enables to exploit the flexibility of our sub-behaviour relations also for sent/received behaviours. In AuthServiceBeh, by

labeling with c the behaviour:

$$(\overline{\texttt{VoteA}}.(\overline{\texttt{Va1}} \oplus \overline{\texttt{Va2}}) \oplus \overline{\texttt{VoteB}}.(\overline{\texttt{Vb1}} \oplus \overline{\texttt{Vb2}}) \oplus \overline{\texttt{VoteC}}.(\overline{\texttt{Vc1}} \oplus \overline{\texttt{Vc2}})) \qquad (1)$$

it is stated that *Vote-Bookkeeper* can interact with any voter willing to express her vote according to any behaviour τ such that

$$(\overline{\texttt{VoteA}}.(\overline{\texttt{Va1}} \oplus \overline{\texttt{Va2}}) \oplus \overline{\texttt{VoteB}}.(\overline{\texttt{Vb1}} \oplus \overline{\texttt{Vb2}}) \oplus \overline{\texttt{VoteC}}.(\overline{\texttt{Vc1}} \oplus \overline{\texttt{Vc2}})) \ \preceq_c \ \tau \qquad (2)$$

For example a possible τ in (2) could be:

$$(\overline{\texttt{VoteA}} \oplus \overline{\texttt{VoteB}})$$

so that, if *Voter3* is committed to behave according to:

$$\mathsf{Voter3Beh} = \overline{\texttt{Login}}.(\texttt{Wrong} + \texttt{Ok}.\texttt{?}[\ (\overline{\texttt{VoteA}} \oplus \overline{\texttt{VoteB}})^{\boldsymbol{C}}\ ])$$

then she is guaranteed to safely interact as a client with the modularized server *Ballot-Service*. In fact by the definition of $\dashv$, we have:

$$\mathsf{Voter3Beh} \dashv \mathsf{AuthServiceBeh}$$

To exemplify sent/received behaviours labelled by s, let us describe the service *Authentication-Service-2*, richer than *Authentication-Service*. Such a service, after having sent the behaviour according to which its client can vote, can receive from the voter a behaviour (a protocol) according to which the voter's employer wishes to be certified about her employee having voted (let us assume that the protocol enables the choice among two sorts of certifications, `C1` or `C2`, and for each of these certifications a certification stamp can be issued, `s1` or `s2`, respectively):

$\mathsf{AuthService2Beh} =$

$$\begin{array}{l}
\texttt{rec}\ x. \\
\ \texttt{Login}. \\
\quad (\overline{\texttt{Wrong}}.\ x \\
\quad \overset{\oplus}{\overline{\texttt{Ok}}}.\ \texttt{!}[\ (\ \overline{\texttt{VoteA}}.(\overset{\oplus}{\overline{\texttt{Va1}}}\ ) \oplus \overline{\texttt{VoteB}}.(\overset{\oplus}{\overline{\texttt{Vb1}}}\ ) \oplus \overline{\texttt{VoteC}}.(\overset{\oplus}{\overline{\texttt{Vc1}}}\ )\ )^{\boldsymbol{C}}\ ].\ \texttt{?(}\ (\texttt{C1}.\overline{\texttt{s1}} + \texttt{C2}.\overline{\texttt{s2}})^{\boldsymbol{S}}\ \texttt{)}\ )
\end{array}$$

Then *Authentication-Service-2* is guaranteed to safely interact with some *Voter4* behaving in accordance with:

$$\mathsf{Voter4Beh} = \overline{\texttt{Login}}.(\texttt{Wrong} + \texttt{Ok}.\texttt{?(}\ (\overline{\texttt{VoteA}} \oplus \overline{\texttt{VoteB}})^{\boldsymbol{C}}\ \texttt{).!}[\ (\texttt{C1})^{\boldsymbol{S}}\ ])$$

since

$$\texttt{C1} \preceq_s (\texttt{C1}.\overline{\texttt{s1}} + \texttt{C2}.\overline{\texttt{s2}}).$$

Behaviour `C1` might look rather meaningless, but the voter could know in advance that the employer will ask for the first certificate, but that she is not waiting for the respective stamp.

It is now natural to expect that, in case one implements an *Authentication-Service-3* according to:

$$\mathsf{AuthService3Beh} = \quad \texttt{rec}\ x.\ \texttt{Login}.(\overline{\texttt{Wrong}}.\ x \oplus \overline{\texttt{Ok}}.\ \texttt{!}[\delta].\texttt{?(}\ (\texttt{C1}.\overline{\texttt{s1}}.\overline{\texttt{d}} + \texttt{C2}.\overline{\texttt{s2}} + \texttt{C3}.\overline{\texttt{s3}})^{s}\ \texttt{)}\ )$$

where C3 denotes another possible certification, d is an extra date-stamp for the certification C1, and δ is (1) (or a greater behaviour w.r.t. $\preceq_c$), we get:

$$\mathsf{AuthService2Beh} \preceq_s \mathsf{AuthService3Beh}$$

This means that, in a system containing AuthService2Beh, such a service can be safely replaced by AuthService3Beh since, for what concerns the higher-order input actions of these behaviours, we have that:

$$(\texttt{C1}.\overline{\texttt{s1}} + \texttt{C2}.\overline{\texttt{s2}}) \preceq_s (\texttt{C1}.\overline{\texttt{s1}}.\overline{\texttt{d}} + \texttt{C2}.\overline{\texttt{s2}} + \texttt{C3}.\overline{\texttt{s3}}).$$

The above example also illustrates the use of the label '$*$'. Let us consider a *Voter5* who behaves the same as *Voter4*, but after sending the login, besides the ability of receiving the information whether the login is ok or not, she can receive a Help message before the description of the interaction with a *PasswordsManager*. The interaction protocol with the *Passwords Manager* allows to retrieve a forgotten password by answering to one of two possible questions: the name of the voter's best friend or the date of the voter's mother birthday.

$$\begin{aligned}\mathsf{Voter5Beh} = \texttt{rec}\ x.\ &\overline{\texttt{Login}}.\\ &(\texttt{Wrong}.x\\ &+\\ &\texttt{Ok.}\,\texttt{![}\ (\overline{\texttt{VoteA}} \oplus \overline{\texttt{VoteB}})^{c}\texttt{]}\\ &+\\ &\texttt{Help.}\,\texttt{?(}\ (\texttt{nameRq}.\overline{\texttt{name}}.\texttt{pw} + \texttt{dateRq}.\overline{\texttt{date}}.\texttt{pw})^{*}\texttt{)})\end{aligned}$$

We assume that the interaction between *Voter5* and *PasswordsManager* cannot be interrupted by any of the two participants. This accounts for the label '$*$' on the received behaviour (nameRq.$\overline{\texttt{name}}$.pw + dateRq.$\overline{\texttt{date}}$.pw). It follows that, given the behaviour:

$$\mathsf{Voter6Beh} = \texttt{rec}\ x.\ \overline{\texttt{Login}}.\ (\texttt{Ok} + \texttt{Help.}\,\texttt{?(}\ (\texttt{nameRq} + \texttt{dateRq}.\overline{\texttt{date}}.\texttt{pw})^{*}\texttt{)})$$

we have that

$$\mathsf{Voter6Beh} \not\preceq_c \mathsf{Voter5Beh}$$

since

$$(\texttt{nameRq} + \texttt{dateRq}.\overline{\texttt{date}}.\texttt{pw}) \not\preceq_* (\texttt{nameRq}.\overline{\texttt{name}}.\texttt{pw} + \texttt{dateRq}.\overline{\texttt{date}}.\texttt{pw})$$

In fact $\overline{\texttt{nameRq}}$ is a peer of (nameRq+dateRq.$\overline{\texttt{date}}$.pw), but not of (nameRq.$\overline{\texttt{name}}$.pw+ dateRq.$\overline{\texttt{date}}$.pw). We have instead:

$$\mathsf{Voter7Beh} \preceq_c \mathsf{Voter5Beh}$$

for

$$\mathsf{Voter7Beh} = \mathtt{rec}\ x.\overline{\mathtt{Login}}.\ (\mathtt{Ok} + \mathtt{Help}.\,\mathtt{?(}\ (\mathtt{dateRq}.\overline{\mathtt{date}}.\mathtt{pw})^{\mathtt{*}}\mathtt{)}).$$

As suggested before, the presence of sent/received behaviours makes the formalization of sub-behaviour relations difficult. In particular because of a circularity problem. As a matter of fact, in the formalization of the interaction between elements of $\mathcal{SB}$, in order to reflect the above discussed variance/contravariance property of input/output sub-behaviour, we should postulate that in case $\sigma \xrightarrow{![\rho_1^p]} \sigma'$ & $\tau \xrightarrow{?(\rho_2^p)} \tau'$, the two interacting behaviours $\sigma\|\tau$ synchronize (formally $\sigma\|\tau \longrightarrow \sigma'\|\tau'$) if and only if $\rho_1 \preceq_p \rho_2$. But $\preceq_p$ depends, via the concept of compliance, on the very same reduction relations, so making the definitions circular.

By analogy with the similar case of Theorem 2.6 in [9], the problem can be solved by means of stratification based on a suitable complexity measure. In fact, in order to perform the input or the output of a ρ^p, any σ necessarily contains ρ as a (proper) subexpression. On passing we observe that this is not true anymore if sent/received behaviors are open expressions, namely if they contain some variable x.

The apparently simple formalization via stratification, however, has both conceptual and technical difficulties. Conceptually, the LTS is not a formal system, at least at first glance, since the logical complexity of the definition is Π_2^0, so not even an r.e. relation. Technically, defining the $\preceq_p$ relations as the union of their stratified restrictions requires a thoroughly study of the LTS before proving that they coincide with the inclusion of the sets $\mathsf{Client}(\sigma)$, $\mathsf{Server}(\rho)$ and $\mathsf{Peer}(\sigma)$, as intended. Such a detailed study, together with the $\preceq_*$ relation properly extend the work in [3].

Once we know that the three sub-behaviour relations over $\mathcal{SB}$ are well defined, their coinductive characterisation is the essential tool for their further investigation. This will be achieved by a mutual coinductive construction, which we shall give in terms of the behaviour syntax up to a notion of convergence, accounting for the unfolding of recursive definitions and of the set of possible internal choices.

A first consequence is the proof that the defined relations $\preceq_s$, $\preceq_c$ and $\preceq_*$ on $\mathcal{SB}$ are well-behaved with respect to duality, as it is induced by the notion of dual session types. Moreover, it will be possible to show that $\preceq_*$ is the intersection of server and client sub-behaviours. A notion of *semantic subtyping* will be also defined as the restriction of the $\preceq_*$ relation to elements of $\mathcal{SB}$ containing only the label $'*$.

We shall also present a client/server/peer subtyping system which simultaneously axiomatises three relations $\leq_c$, $\leq_s$ and $\leq_*$ on session types, transparently corresponding to $\preceq_c$, $\preceq_s$ and $\preceq_*$. The type syntax reflects the definition of $\mathcal{SB}$ in that input/output types have the transmitted type A_1 in $?(A_1^p)A_2$ and in $![A_1^p]A_2$ labeled by a polarity. We shall require A_1 to be a closed type, a necessary restriction which was not taken into account in [16]. Then, to the unfolding rules of recursive μ-types and the coinductive rules treating each type constructor but μ, we add the axioms $A \leq_c \mathsf{end}$, $\mathsf{end} \leq_s A$ and $\mathsf{end} \leq_* \mathsf{end}$ for any A, the first two accounting for the asymmetry of the client and server sub-behaviour relations.

The typing system defines a set of derivable judgments $\Gamma \vdash A \leq_p B$, where

$\Gamma \cup \{A \leq_p B\}$ is a set of inequalities among closed types. Its semantic counterpart is $\Gamma \models A \leq_p B$, which is defined as the usual conditional statement that if all the inequalities $C \leq_q D \in \Gamma$ are true according to the definition $\models C \leq_q D \Leftrightarrow [\![C]\!] \preceq_q [\![D]\!]$, then $\models A \leq_p B$. The soundness theorem states that $\Gamma \vdash A \leq_p B$ implies $\Gamma \models A \leq_p B$; on the other hand the opposite implication fails when $\not\models \Gamma$, as it will be shown by a counterexample. In fact the best result one can establish is that if $\Gamma \models A \leq_p B$ then either $\not\models \Gamma$ or $\Gamma \vdash A \leq_p B$. But this is enough to show that $[\![A]\!] \preceq_p [\![B]\!]$ if and only if $\emptyset \vdash A \leq_p B$, for $p = c, s, *$. Such a completeness result extends and completes the semantic investigation started in [3], where just soundness was proved. A soundness and completeness result is shown to hold also for Gay and Hole's subtyping w.r.t. our semantic subtyping. Moreover, since our subtyping system is shown to be decidable and behaviours are bijective with session types, we also deduce from the completeness theorem that the relations $\preceq_c$, $\preceq_s$ and $\preceq_*$ are decidable, and so that the LTS defining the operational semantics of behaviours is a formal system in the usual sense.

The paper is organised as follows: in section 2 we introduce session behaviours, the LTS defining compliance and hence the server, client and peer sub-behavior relations, stratified w.r.t. the rank of behaviour expressions. We prove that the stratified definition induces three chains of relations, whose unions coincide with the inclusion of clients, servers and peers of each pair of related behaviours, respectively. In section 3 we provide a coinductive characterisation of the sub-behaviour relations, which is entirely based on the syntax of behaviour expressions, not on the LTS. By means of the coinductive characterization we shall be able to prove the apparently simple and natural property that $\preceq_*$ is the intersection of $\preceq_c$ and $\preceq_s$. In Section 4 we define session types with polarized higher-order input/output, dubbed CSP-session types, and a derivation system axiomatizing client/server/peer subtyping (that we prove to be decidable), which includes the Gay and Hole subtyping for session types. We then interpret in Section 5 CSP-session types into session behaviours and client/server/peer subtyping into the respective sub-behaviour relations, proving soundness and completeness of the system. We derive also that a suitable restriction of $\preceq_*$ (the semantic subtyping) is a model for the Gay-Hole subtyping relation. In section 6 we refer to related works, and in section 7 we conclude.

The present paper extends the systems and results of the preliminary paper [3], where only the client/server relations were taken into account and where no full proof were provided. Here we also provide a complete analysis of the stratified definition of the sub-behaviour relations. Besides, in [3], only soundness was shown for our subtyping system. The present paper also provides the proof of decidability for the client/server/peer subtype relations,implying, by completeness, the decidability of the sub-behaviour relations.

2 Session Behaviours and Client/Server Sub-Behaviour relations

A *session behaviour* can be looked at as an abstract description of the communication actions by a process on one end of a bidirectional channel (or, from a more general viewpoint, as an abstract description of the behaviour of a com-

ponent) in a distributed system. We formalize session behaviours by means of a process calculus inspired by the calculus of contracts [7, 18, 11, 10] which, on the one hand, we restrict w.r.t. the shape of subterms of both external and internal choices, while, on the other hand, we extend contract syntax to model delegation (higher-order input and output.)

Definition 2.1 (Session Behaviours) *i) Let $\mathcal{N}$ be some countable set of symbols and $\overline{\mathcal{N}} = \{\overline{a} \mid a \in \mathcal{N}\}$, with $\mathcal{N} \cap \overline{\mathcal{N}} = \emptyset$. The set $\mathcal{BE}$ of* raw behaviour expressions *is defined by the grammar:*

$$
\begin{array}{rcll}
\sigma, \tau & ::= & \mathbf{1} & \textit{inaction} \\
& \mid & a_1.\sigma_1 + \cdots + a_n.\sigma_n & \textit{external choice} \\
& \mid & \overline{a}_1.\sigma_1 \oplus \cdots \oplus \overline{a}_n.\sigma_n & \textit{internal choice} \\
& \mid & x & \textit{variable} \\
& \mid & \mathsf{rec}\, x.\sigma & \textit{recursion} \\
& \mid & ?(\sigma^p)\tau & \textit{input} \qquad p \in \{s, c, *\} \\
& \mid & ![\sigma^p]\tau & \textit{output}
\end{array}
$$

where

- *$n \geq 1$ and $a_i \in \mathcal{N}$ (hence $\overline{a}_i \in \overline{\mathcal{N}}$) for all $1 \leq i \leq n$;*
- *x is a session behaviour variable out of a denumerable set and it is bound by the* rec *operator;* $\textsc{fv}(\sigma)$ *denotes, as usual, the set of free variables in σ.*

ii) The set $\mathcal{SB}$ of session behaviours *is the subset of closed raw behaviour expressions such that:*

- *in $a_1.\sigma_1 + \cdots + a_n.\sigma_n$ and $\overline{a}_1.\sigma_1 \oplus \cdots \oplus \overline{a}_n.\sigma_n$, the a_i and $\overline{a}_i$ are, respectively, pairwise distinct;*
- *in $\mathsf{rec}\, x.\sigma$ the expression σ is not a variable;*
- *in $?(\sigma^p)\tau$ and $![\sigma^p]\tau$ σ is a* **closed** *expression, i.e.* $\textsc{fv}(\sigma) = \emptyset$.

We abbreviate $a_1.\sigma_1 + \cdots + a_n.\sigma_n$ by $\sum_{i=1}^n a_i.\sigma_i$, and $\overline{a}_1.\sigma_1 \oplus \cdots \oplus \overline{a}_n.\sigma_n$ by $\bigoplus_{i=1}^n \overline{a}_i.\sigma_i$. We also use the notations $\sum_{i\in I} a_i.\sigma_i$ and $\bigoplus_{i\in I} \overline{a}_i.\sigma_i$, for finite and not empty I. The trailing $\mathbf{1}$ is normally omitted: we write e.g. $a+b$ for $a.\mathbf{1}+b.\mathbf{1}$. Note that recursion in $\mathcal{SB}$ is guarded and hence contractive in the usual sense.

This definition follows similar constructions in [8, 21]. A technical difference is the use of polarities; they have been introduced in [16] and used in [23] to keep track of the pairing of the two ends of private channels of sessions. We use here polarities to distinguish among actions by a server ($p = s$), by a client ($p = c$), or by a peer ($p = *$).

We comment on the restriction that $\textsc{fv}(\sigma) = \emptyset$ in $?(\sigma^p)\tau$ and $![\sigma^p]\tau$ in Remark 2.4.

The semantics of session behaviours is given in terms of a labeled transition system (LTS) $\sigma \xrightarrow{\alpha} \sigma'$ where $\sigma, \sigma' \in \mathcal{SB}$ and α belongs to an appropriate set of actions: **Act**. As usual, $\mathcal{N} \cup \overline{\mathcal{N}} \subseteq \mathbf{Act}$; in the present case, however, also the input/output of a behaviour is an action so that, somehow, $\mathcal{SB}$ has to be included into **Act**.

Definition 2.2 (Behaviour LTS)
Define the set of actions $\mathbf{Act}_0 = \mathcal{N} \cup \overline{\mathcal{N}}$ *and*

$$\mathbf{Act} = \mathbf{Act}_0 \cup \{?(\sigma^p), ![\sigma^p] \mid \sigma \in \mathcal{SB}, p \in \{s, c, *\}\}.$$

Let $\oplus, \mathsf{rec} \notin \mathbf{Act}$*; define the LTS* $(\mathcal{SB}, \mathbf{Act} \cup \{\oplus, \mathsf{rec}\}, \longrightarrow)$ *by the rules:*

$$a_1.\sigma_1 + \cdots + a_n.\sigma_n \xrightarrow{a_k} \sigma_k \qquad \overline{a}.\sigma \xrightarrow{\overline{a}} \sigma$$

$$\overline{a}_1.\sigma_1 \oplus \cdots \oplus \overline{a}_n.\sigma_n \xrightarrow{\oplus} \overline{a}_k.\sigma_k \qquad \mathsf{rec}\, x.\sigma \xrightarrow{\mathsf{rec}} \sigma\{\mathsf{rec}\, x.\sigma/x\}$$

$$?(\sigma^p)\tau \xrightarrow{?(\sigma^p)} \tau \qquad ![\sigma^p]\tau \xrightarrow{![\sigma^p]} \tau$$

where $1 \le k \le n$ *and* $\sigma \xrightarrow{\alpha} \tau$ *abbreviates* $(\sigma, \alpha, \tau) \in \longrightarrow$.

We abbreviate $\longrightarrow = \xrightarrow{\oplus} \cup \xrightarrow{\mathsf{rec}}$. Note that neither $\oplus$ nor rec are actions, so that they are unobservable and used just for technical reasons; indeed we adopt the standard $\longrightarrow$ (from CCS without τ) in the subsequent definition of the parallel operator for testing.

As usual, we write $\Longrightarrow = \longrightarrow^*$, $\overset{\alpha}{\Longrightarrow} = \longrightarrow^* \xrightarrow{\alpha} \longrightarrow^*$ for $\alpha \in \mathbf{Act}$, $\sigma \overset{s}{\Longrightarrow} \sigma'$ if $s = \alpha_1 \cdots \alpha_n$ and $\sigma \overset{\alpha_1}{\Longrightarrow} \cdots \overset{\alpha_n}{\Longrightarrow} \sigma'$. Also we write $\sigma \longrightarrow$ and $\sigma \xrightarrow{\alpha}$ if there exists σ' s.t. $\sigma \longrightarrow \sigma'$ and $\sigma \xrightarrow{\alpha} \sigma'$ respectively, and $\sigma \not\longrightarrow$ when $\neg(\sigma \longrightarrow)$.

A syntactical concept of *duality* on $\mathcal{SB}$ is obtained by interchanging a with $\overline{a}$, $+$ with $\oplus$ and $?(\cdot)$ with $![\cdot]$. Its formal definition is obtained by restricting to $\mathcal{SB}$ a straightforward definition by induction on the structure of the raw expressions in $\mathcal{BE}$ (i.e. also for open expressions).

From now on, in order to avoid too cumbersome definitions, any time an inductive definition on elements of $\mathcal{SB}$ is provided, it will be tacitely assumend to be actually the restriction to $\mathcal{SB}$ of the corresponding inductive definition on $\mathcal{BE}$.

Definition 2.3 *The syntactic dual* $\overline{\sigma}$ *of* $\sigma \in \mathcal{SB}$ *is defined by the following clauses:*

$$\overline{\mathbf{1}} = \mathbf{1} \qquad \overline{x} = x \qquad \overline{\mathsf{rec}\, x.\sigma} = \mathsf{rec}\, x.\overline{\sigma}$$

$$\overline{\textstyle\sum_{i\in I} a_i.\sigma_i} = \bigoplus_{i\in I} \overline{a}_i.\overline{\sigma_i} \qquad \overline{\bigoplus_{i\in I} \overline{a}_i.\sigma_i} = \textstyle\sum_{i\in I} a_i.\overline{\sigma_i}$$

$$\overline{?(\sigma^p)\tau} = ![\sigma^p]\overline{\tau} \qquad \overline{![\sigma^p]\tau} = ?(\sigma^p)\overline{\tau}$$

The definition closely mimics the duality operator on session types, e.g. in [16]. As expected, $\overline{\overline{\sigma}} = \sigma$ for all σ. Note that the behaviour σ in $\overline{?(\sigma^p)\tau}$, as well as its polarity p remains unaffected by the $\bar{\cdot}$ operation, and similarly for $\overline{![\sigma^p]\tau}$.

Remark 2.4 *By the fact that* rec *is not observable and that* $\mathsf{rec}\, x.\sigma \xrightarrow{\mathsf{rec}} \sigma\{\mathsf{rec}\, x.\sigma/x\}$ *is the unique possible reduction out of* $\mathsf{rec}\, x.\sigma$*, the behaviours* $\mathsf{rec}\, x.\sigma$ *and* $\overline{\sigma\{\mathsf{rec}\, x.\sigma/x\}}$ *are observationally indistiguischable. So we expect that both* $\overline{\mathsf{rec}\, x.\sigma}$ *and* $\overline{\sigma\{\mathsf{rec}\, x.\sigma/x\}}$ *are such. But this is false without the restriction that* $\mathrm{FV}(\sigma) = \emptyset$ *in* $?(\sigma^p)\tau$ *and* $![\sigma^p]\tau$*. Consider for example (with any* p*):*

$$\mathsf{rec}\, x.?(x^p) \xrightarrow{\mathsf{rec}} ?(\mathsf{rec}\, x.?(x^p)^p)$$

where $\mathsf{rec}\, x.?(x^p)$ *violates the constraint on input/output behaviors because, even if it is closed, the* x *is free in the subexpression* $?(x^p)$*. Then we have:*

$$\overline{\mathsf{rec}\, x.?(x^p)} = \mathsf{rec}\, x.![x^p] \xrightarrow{\mathsf{rec}} ![\mathsf{rec}\, x.![x^p]^p].$$

But $\overline{?(\mathsf{rec}\, x.?(x^p)^p)} = ![\mathsf{rec}\, x.?(x^p)^p] \neq ![\mathsf{rec}\, x.![x^p]^p]$.

2.1 Defining Compliance and sub-Behaviours Relations by Stratification

As discussed in the Introduction, we intend to formalize three binary sub-behaviour relations on $\mathcal{SB}$ that represent the notion of substitutability in client/server or peer systems and that take into account the different roles of the interacting components in such systems. The relation of *client sub-behaviour*, $\rho_1 \preceq_c \rho_2$, is defined as to the the subset relation among the possible *servers* of ρ_1 and ρ_2; the relation of *server sub-behaviour*, $\sigma_1 \preceq_s \sigma_2$, instead, is the subset relation among the possible *clients* of σ_1 and σ_2; finally, the relation of *peer sub-behaviour*, $\sigma_1 \preceq_* \sigma_2$ is the subset relation among the possible *peers* of σ_1 and σ_2.

All the notions of *client*, *server* and *peer* are intended to be formalized by introducing a relation of *compliance* holding between two elements of $\mathcal{SB}$, $\rho \dashv \sigma$, whenever any action α on ρ's side (α can be looked at as a *client request*) is eventually matched by a correspondent synchronizing co-action $\tilde{\alpha}$ on σ's side (then $\tilde{\alpha}$ is a *server response*). We shall see that, as exemplified in the examples of the Introduction, $\tilde{\alpha}$ does not need to be the syntactic dual of α in case of higher-order actions.

As previously mentioned, and further discussed later on, the presence of higher-order actions makes the definition of action synchronization, and hence that of the relation $\dashv$, depend on all $\preceq_c$, $\preceq_s$, and $\preceq_*$, so revealing a circularity.

However, thanks to the restriction that input/output behaviors cannot include any free occurrence of variables, we can argue that the σ in $?(\sigma^p).\tau$ and $![\sigma^p].\tau$ is always of lower complexity than the whole behavior expression, so that we can stratify the definition of $\mathcal{SB}$ w.r.t. the following notion of *rank*.

Definition 2.5 (Stratified session behaviours) *Let us define a* rank *mapping* $\mathsf{rank} : \mathcal{SB} \to \mathbb{N}$ *inductively as follows:*

$$\begin{array}{rcl}
\mathsf{rank}(\mathbf{1}) & = & \mathsf{rank}(x) = 0 \\
\mathsf{rank}(\mathsf{rec}\, x.\sigma) & = & \mathsf{rank}(\sigma) \\
\mathsf{rank}(\sum_{i=1}^n a_i.\sigma_i) = \mathsf{rank}(\bigoplus_{i=1}^n \overline{a}_i.\sigma_i) & = & \max(\mathsf{rank}(\sigma_1), \ldots, \mathsf{rank}(\sigma_n)) \\
\mathsf{rank}(?(\sigma^p)\tau) = \mathsf{rank}(![\sigma^p]\tau) & = & \max(\mathsf{rank}(\sigma) + 1, \mathsf{rank}(\tau))
\end{array}$$

Then we set $\mathcal{SB}_i = \{\sigma \in \mathcal{SB} \mid \mathsf{rank}(\sigma) \leq i\}$.

The number $\mathsf{rank}(\sigma)$ measures the maximal nesting of input/output actions in σ. Since we forbid the input/output of open behaviours, we have that $\mathsf{rank}(\mathsf{rec}\, x.\sigma) = \mathsf{rank}(\sigma\{\mathsf{rec}\, x.\sigma/x\})$ for all raw behavioural expressions σ. This is the consequence of the following lemma and justifies the definition $\mathsf{rank}(x) = 0$.

Lemma 2.6 *For all $\sigma, \tau \in \mathcal{SB}$ and variable x we have:*

$$\mathsf{rank}(\sigma\{\tau/x\}) = \begin{cases} \max(\mathsf{rank}(\sigma), \mathsf{rank}(\tau)) & \text{if } x \in \mathrm{FV}(\sigma) \\ \mathsf{rank}(\sigma) & \text{otherwise.} \end{cases}$$

It follows that $\mathsf{rank}(\mathsf{rec}\, x.\sigma) = \mathsf{rank}(\sigma\{\mathsf{rec}\, x.\sigma/x\})$ *for any* σ.

Proof. If $x \notin \mathrm{FV}(\sigma)$ then obviously $\sigma\{\tau/x\} = \sigma$ and the thesis is trivial. Otherwise we reason by induction over the structure of σ. If $x \in \mathrm{FV}(\sigma)$ then σ cannot be $\mathbf{1}$ nor any variable different from x; if $\sigma = x$ then $\sigma\{\tau/x\} = \tau$ and $\mathsf{rank}(\tau) \geq \mathsf{rank}(x) = 0$. If σ is either an internal or an external choice the thesis follows by the induction hypothesis. The relevant cases are when $\sigma = ?(\sigma_0^p)\sigma_1$ or $\sigma = ![\sigma_0^p]\sigma_1$. Then e.g. $(?(\sigma_0^p)\sigma_1)\{\tau/x\} = ?(\sigma_0^p)(\sigma_1\{\tau/x\})$ because $x \notin \mathrm{FV}(\sigma_0)$, being σ_0 closed by Definition 2.1. So the thesis follows by the induction hypothesis.

Now $\mathsf{rank}(\mathsf{rec}\, x.\sigma) = \mathsf{rank}(\sigma)$, so that $\mathsf{rank}(\mathsf{rec}\, x.\sigma) = \mathsf{rank}(\sigma\{\mathsf{rec}\, x.\sigma/x\})$ no matter whether $x \in \mathrm{FV}(\sigma)$ or not. □

If $\alpha \in \mathbf{Act}$, by abuse of notation we write $\mathsf{rank}(\alpha) = 0$ if $\alpha \in \mathcal{N} \cup \overline{\mathcal{N}}$ and $\mathsf{rank}(?(\sigma^p)) = \mathsf{rank}(![\sigma^p]) = \mathsf{rank}(\sigma) + 1$.

We now formally introduce the relation of *compliance* for session behaviours of rank 0, written $\rho \dashv_0 \sigma$. Then, by means of the compliance relation, we define the sets of clients, servers and peers of level 0, and finally the sub-behaviour relations of level 0.

We recall that behaviours in $\mathcal{SB}_0$ do not contain $?(\sigma^p)\tau$ nor $![\sigma^p]\tau$.

Definition 2.7 (Compliance and Sub-behaviours for $\mathcal{SB}_0$) *i) Let $\rho \| \sigma$ denote a pair of session behaviors in $\mathcal{SB}_0$, then define:*

$$\frac{\rho \xrightarrow{\alpha} \rho'}{\rho \xrightarrow{\alpha}_0 \rho'} \qquad \frac{\rho \longrightarrow \rho'}{\rho \longrightarrow_0 \rho'}$$

$$\frac{\rho \xrightarrow{\alpha}_0 \rho' \quad \sigma \xrightarrow{\overline{\alpha}}_0 \sigma'}{\rho \| \sigma \longrightarrow_0 \rho' \| \sigma'} \qquad \frac{\rho \longrightarrow_0 \rho'}{\rho \| \sigma \longrightarrow_0 \rho' \| \sigma} \qquad \frac{\sigma \longrightarrow_0 \sigma'}{\rho \| \sigma \longrightarrow_0 \rho \| \sigma'}$$

where $\alpha \in \mathcal{N} \cup \overline{\mathcal{N}}$ and $\overline{\alpha}$ is the usual involution: $\overline{\overline{\alpha}} = \alpha$.

ii) We say that the client *ρ is* compliant *with the* server *σ, written $\rho \dashv_0 \sigma$, if*

$$\forall \rho', \sigma'.\ \rho \| \sigma \longrightarrow_0^* \rho' \| \sigma' \not\longrightarrow_0 \quad \Rightarrow \quad \rho' = \mathbf{1}$$

The session behaviour ρ is said to be a peer *of the behaviour σ, written $\rho \perp_0 \sigma$, if both $\rho \dashv_0 \sigma$ and $\sigma \dashv_0 \rho$ hold.*

iii) We define:

$$\mathsf{Client}_0(\sigma) = \{\rho \in \mathcal{SB}_0 \mid \rho \dashv_0 \sigma\} \qquad \mathsf{Server}_0(\rho) = \{\sigma \in \mathcal{SB}_0 \mid \rho \dashv_0 \sigma\}$$

$$\mathsf{Peer}_0(\sigma) = \{\rho \in \mathcal{SB}_0 \mid \rho \perp_0 \sigma\}.$$

iv) We define the following sub-behaviour relations over $\mathcal{SB}_0$:

$$\begin{aligned}
\sigma \preceq^0_s \sigma' \quad &\Leftrightarrow \quad \emptyset \neq \mathsf{Client}_0(\sigma) \subseteq \mathsf{Client}_0(\sigma'),\\
\rho \preceq^0_c \rho' \quad &\Leftrightarrow \quad \emptyset \neq \mathsf{Server}_0(\rho) \subseteq \mathsf{Server}_0(\rho'),\\
\sigma \preceq^0_* \sigma' \quad &\Leftrightarrow \quad \emptyset \neq \mathsf{Peer}_0(\sigma) \subseteq \mathsf{Peer}_0(\sigma').
\end{aligned}$$

Notice that by restricting the attention to behaviours in $\mathcal{SB}_0$, the compliance relation is defined as in [18, 20].

Simple examples of compliance are: $\overline{a}\oplus\overline{b} \dashv_0 a.d+b$ and also $\overline{a}\oplus\overline{b} \dashv_0 a.d+b+e$, whereas $\overline{a} \oplus \overline{b} \perp_0 a+b$; on the other hand $a+b \not\dashv_0 \overline{a} \oplus \overline{d}$ because the behaviour $\overline{a} \oplus \overline{d}$ could reduce to $\overline{d}$. This would prevent a possible request of $a+b$, the one modeled by the action $\xrightarrow{a}$, from being matched by a corresponding action on $\overline{d}$'s side.

The $\dashv_0$ relation is not symmetric: indeed $\mathbf{1} \dashv_0 \sigma$ for any σ, but e.g. $a \not\dashv_0 \mathbf{1}$. Observe that in [18, 20] $a \oplus b \not\dashv_0 \overline{a} \oplus \overline{b}$ while $a+b \dashv_0 \overline{a}+\overline{b}$. On the other hand, by our syntactical restrictions neither $a \oplus b$ nor $\overline{a}+\overline{b}$ are well formed session behaviours.

We stress that in our formalism two behaviours can be compliant, or peers, without necessarily exhibiting a finite sequence of syncronising actions: the simplest example is $\mathsf{rec}\, x.a.x \dashv_0 \mathsf{rec}\, x.\overline{a}.x$.

As recalled at the beginning of this subsection, if we wish to have a flexible and expressive formalism, the extension of the definition of compliance to the whole $\mathcal{SB}$ is not straightforward. In fact to cope with higher-order input/output behaviours, the following rule would be sound but unnecessarily restrictive:

$$\frac{\sigma \xrightarrow{![\rho^p]} \sigma' \quad \tau \xrightarrow{?(\rho^p)} \tau'}{\sigma \| \tau \longrightarrow \sigma' \| \tau'} \tag{3}$$

For istance, it would prevent, in the example of the Introduction, the behaviour Voter3Beh to comply with AuthServiceBeh. It would not guarantee as well the possibility of safely substitute AuthService2Beh by AuthService3Beh.

As a matter of fact, given that one is able to extend the sub-behaviour relations $\preceq^0_s$, $\preceq^0_c$ and $\preceq^0_*$ to $\mathcal{SB}$, it would be reasonable to relax the above rule (3) as follows:

$$\frac{\sigma \xrightarrow{![\rho_1^p]} \sigma' \quad \tau \xrightarrow{?(\rho_2^p)} \tau' \quad \rho_1 \preceq_p \rho_2}{\sigma \| \tau \longrightarrow \sigma' \| \tau'} \tag{4}$$

where $p = s, c, *$.

The intuitive justification of rule (4) above is as follows: let us look at a received behaviour as a protocol that the receiving agents is expected to conform to. The superscript polarity specifies the role to play among client, server or peer when the protocol is used. Without specifying the polarity, rule (4) either reduces to rule (3) or to a version of (4) where only $\preceq_*$ can be considered. But this would be a remarkable loss of flexibility. Indeed an agent could keep on safely interacting with the environment even in case of waiting for a client protocol ρ_2 and receiving a *less demanding* client protocol ρ_1 (that is $\rho_1 \preceq_c \rho_2$).

Dually, the interaction with the environment cannot produce undesired effects if a *richer* server protocol ρ_1 is received instead of the expected and poorer ρ_2 (that is $\rho_1 \preceq_s \rho_2$). A similar explanation can be given for the $\preceq_*$ relation.

Rule (4) above, however, cannot be used to define the relation $\longrightarrow$, since it would make the definition circular: in fact by using such a rule the relation $\longrightarrow$ relies on the definition of $\preceq_p$, that is defined in terms of $\dashv$, which in turn is defined in terms of $\longrightarrow$ itself.

Fortunately, as it is suggested in [10, 21], if $\sigma \xrightarrow{?(\rho^p)} \sigma'$ or $\sigma \xrightarrow{![\rho^p]} \sigma'$ then $\mathsf{rank}(\rho) < \mathsf{rank}(?(\rho^p)) = \mathsf{rank}(![\rho^p]) \leq \mathsf{rank}(\sigma)$, so that the circularity can be avoided by *stratifying* the definitions.

Below we consider stratified notions of client/server/peer sub-behaviour.

Definition 2.8 (Compliance and stratified Sub-behaviours) *For each $i \in \mathbb{N}$ we define inductively the binary relations $\preceq_c^i$, $\preceq_s^i$ and $\preceq_*^i$. The cases of $\preceq_c^0$, $\preceq_s^0$ and $\preceq_*^0$ have been treated in Definition 2.7. When $i > 0$ let us add to the rules in the definition of $\longrightarrow_0$ the following one:*

$$\frac{\tau \xrightarrow{![\rho_1^p]} \tau' \quad \sigma \xrightarrow{?(\rho_2^p)} \sigma' \quad \rho_1 \preceq_p^k \rho_2 \quad k = \max(\mathsf{rank}(\rho_1), \mathsf{rank}(\rho_2))}{\sigma \| \tau \longrightarrow \sigma' \| \tau'} \tag{5}$$

and its symmetric, where:

i) $\rho \dashv \sigma \quad \Leftrightarrow \quad \forall \rho', \sigma' \in \mathcal{SB}.\ \ \rho \| \sigma \longrightarrow^* \rho' \| \sigma' \not\longrightarrow \quad \Rightarrow \quad \rho' = \mathbf{1}$;

ii) $\rho \perp \sigma \quad \Leftrightarrow \quad \rho \dashv \sigma \ \ \& \ \ \sigma \dashv \rho$;

iii) $\mathsf{Client}_i(\sigma) = \begin{cases} \{\rho \in \mathcal{SB}_i \mid \rho \dashv \sigma\} & \text{if } \mathsf{rank}(\sigma) \leq i \\ \emptyset & \text{otherwise} \end{cases}$

iv) $\mathsf{Server}_i(\rho) = \begin{cases} \{\sigma \in \mathcal{SB}_i \mid \rho \dashv \sigma\} & \text{if } \mathsf{rank}(\rho) \leq i \\ \emptyset & \text{otherwise} \end{cases}$

v) $\mathsf{Peer}_i(\sigma) = \begin{cases} \{\rho \in \mathcal{SB}_i \mid \rho \perp \sigma\} & \text{if } \mathsf{rank}(\sigma) \leq i \\ \emptyset & \text{otherwise} \end{cases}$

vi) $\sigma \preceq_s^i \sigma' \ \Leftrightarrow \ \emptyset \neq \mathsf{Client}_i(\sigma) \subseteq \mathsf{Client}_i(\sigma')$;

vii) $\rho \preceq_c^i \rho' \ \Leftrightarrow \ \emptyset \neq \mathsf{Server}_i(\rho) \subseteq \mathsf{Server}_i(\rho')$;

viii) $\rho \preceq_*^i \rho' \ \Leftrightarrow \ \emptyset \neq \mathsf{Peer}_i(\rho) \subseteq \mathsf{Peer}_i(\rho')$.

Also for $\longrightarrow$ we use the standard denotation $\Longrightarrow = \longrightarrow^*$.

The following lemma easily descends from the definitions.

Lemma 2.9

i) $\longrightarrow$ *preserves duality; that is* $\quad \overline{\sigma}\|\sigma \longrightarrow \sigma_1\|\sigma_2 \quad \Rightarrow \quad \sigma_2 \equiv \overline{\sigma_1}$.

ii) $\rho \perp \sigma \quad \Leftrightarrow \quad \forall \rho', \sigma' \in \mathcal{SB}.\ \ \rho\|\sigma \longrightarrow^* \rho'\|\sigma' \not\longrightarrow \quad \Rightarrow \quad \rho' = \mathbf{1} \ \ \& \ \ \sigma' = \mathbf{1}$;

iii) For all $i \in \mathbb{N}$, $\mathsf{Peer}_i(\sigma) = \mathsf{Server}_i(\sigma) \cap \mathsf{Client}_i(\sigma)$.

Notice that we immediately have $\preceq^i_* \supseteq \preceq^i_c \cap \preceq^i_s$. More work is needed to establish the desired $\preceq^i_* = \preceq^i_c \cap \preceq^i_s$.

Lemma 2.10

i) $\sigma \xrightarrow{\alpha} \sigma' \ \vee \ \sigma \longrightarrow \sigma' \quad \Rightarrow \quad \mathsf{rank}(\sigma) \geq \mathsf{rank}(\sigma')$;

ii) $\sigma \xrightarrow{?(\rho^p)} \sigma' \ \vee \ \sigma \xrightarrow{![\rho^p]} \sigma' \quad \Rightarrow \quad \mathsf{rank}(\sigma) > \mathsf{rank}(\rho)$;

iii) $\rho\|\sigma \longrightarrow \rho'\|\sigma' \quad \Rightarrow \quad \mathsf{rank}(\rho) \geq \mathsf{rank}(\rho') \ \ \& \ \ \mathsf{rank}(\sigma) \geq \mathsf{rank}(\sigma')$.

Proof. All the three points are quite immediate. In particular point (iii) follows by points (i) and (ii) and does not depend on the premise $\preceq^k_p$ in Def. 2.8. □

Proposition 2.11 *For all* $i \in \mathbb{N}$ *the relations* $\preceq^i_s$, $\preceq^i_c$ *and* $\preceq^i_*$, *as well as the sets* $\mathsf{Client}_i(\sigma)$, $\mathsf{Server}_i(\rho)$ *and* $\mathsf{Peer}_i(\rho)$, *are well defined.*

Proof. By simultaneous induction on i. The base case being obvious, we treat the induction case for $\preceq^i_s$ only, since the other ones are either implied or similar. For $\sigma \preceq^i_s \sigma'$ to be well defined, this has to be the case for both $\mathsf{Client}_i(\sigma)$ and $\mathsf{Client}_i(\sigma')$, that is for any $\rho \in \mathcal{SB}_i$ the statements $\rho \dashv \sigma$ and $\rho \dashv \sigma'$ must be defined. This requires that also $\sigma, \sigma' \in \mathcal{SB}_i$, so that by (iii) of Lemma 2.10, if $\rho\|\sigma \Longrightarrow \rho'\|\sigma''$ or $\rho\|\sigma' \Longrightarrow \rho'\|\sigma''$ we have that $\mathsf{rank}(\rho'), \mathsf{rank}(\sigma'') \leq i$. It follows that, if anywhere in these reductions an instance of rule (5) in Def. 2.8 ever occurs, the rank k in the premise is strictly lower than i by part (ii) of Lemma 2.10, so that the relation $\preceq^k_p$ in the premise is well defined by the inductive hypothesis. □

We can now define the client/server/peer sub-behaviour relations.

Definition 2.12 (Server/Client/Peer Sub-behavior Relations) *Over* $\mathcal{SB}$ *we define the binary relations:*

$$\preceq_s = \bigcup_{i \in \mathbb{N}} \preceq^i_s \qquad \preceq_c = \bigcup_{i \in \mathbb{N}} \preceq^i_c \qquad \preceq_* = \bigcup_{i \in \mathbb{N}} \preceq^i_* .$$

From the above definition it is not immediate to get a clear a picture of what are the properties of the sub-behaviour relations. In particular of what is the relationship among $\preceq_*$, $\preceq_c$ and $\preceq_s$, for which a thoroughly analysis of the stratified definitions will be necessary.

We start by showing that the sub-behaviour relations are indeed reflexive.

Lemma 2.13 *Let $\sigma \in \mathcal{SB}$ with $k = \mathsf{rank}(\sigma)$, and $p = s, c, *$.*

i) $\overline{\sigma} \in \mathsf{Client}_k(\sigma) \cap \mathsf{Server}_k(\sigma)$*;*

ii) $\sigma \preceq^k_p \sigma$*.*

Proof. We prove (i) and (ii) by simultaneous induction over k. For (i) we need to consider just $\overline{\sigma} \in \mathsf{Client}_k(\sigma) \cap \mathsf{Server}_k(\sigma)$, since $\overline{\sigma} \in \mathsf{Peer}_k(\sigma)$ will immediately descends from Lemma 2.9(iii).
Case $k = 0$. The theses descend immediately from Definition 2.7.
Case $k > 0$. We consider (i) first. We show only that $\overline{\sigma} \in \mathsf{Client}_k(\sigma)$, being the proof for $\overline{\sigma} \in \mathsf{Server}_k(\sigma)$ similar.
Then, by definition of $\mathsf{Client}_k(\sigma)$ we have to show, taking into account Lemma 2.9(i), that $\forall \sigma' \in \mathcal{SB}.\ \overline{\sigma} \| \sigma \Longrightarrow \overline{\sigma'} \| \sigma' \not\longrightarrow \Rightarrow\ \overline{\sigma'} = \mathbf{1}$. This can be shown by using the definition of $\longrightarrow$ and the definition of syntactic duality (Def. 2.3). In case of higher-order actions we can apply the induction hypothesis for (ii). In fact, whenever rule (5) has to be used in the derivation, we need to have $\rho' \preceq^h_p \rho'$ for some ρ', with $h = \mathsf{rank}(\rho')$. This is exactly the induction hypothesis of (ii), since, by Lemma 2.10(iii) and definition of rank, we get $\mathsf{rank}(\rho') < \mathsf{rank}(\sigma)$, that is $h < k$.
We can now proceed with (ii). By point (i) we have that $\mathsf{Client}_k(\sigma) \neq \emptyset$ and $\mathsf{Server}_k(\sigma) \neq \emptyset$. Then, by definition of $\preceq^k_p$, what we need to show is just the trivial facts that $\mathsf{Client}_k(\sigma) \subseteq \mathsf{Client}_k(\sigma)$ and $\mathsf{Server}_k(\sigma) \subseteq \mathsf{Server}_k(\sigma)$. □

Observe that, by the last lemma, conditions $\emptyset \neq \mathsf{Client}_i(\sigma)$, $\emptyset \neq \mathsf{Server}_i(\sigma)$ and $\emptyset \neq \mathsf{Peer}_i(\sigma)$ in Definition 2.8 are equivalent to $\max(\mathsf{rank}(\sigma), \mathsf{rank}(\sigma')) \leq i$.

A relevant property of the sub-behaviour relations is illustrated by the following proposition which roughly says that $\preceq_s$ has a bottom element and $\preceq_c$ a top one, both coinciding with $\mathbf{1}$. When we define client/server/peer subtyping relations for session types in Section 5 (Def. 4.2), these properties will be explicitly represented by the axioms (T-Ax-C) and (T-Ax-S), so that the client/server subtypings is sound and complete w.r.t. the corresponding subbehaviour relations.

Observe that $\mathsf{Client}_i(\mathbf{1}) \neq \{\mathbf{1}\}$ since e.g. $\mathsf{rec}\, x.\mathbf{1} \in \mathsf{Client}_i(\mathbf{1})$; on the other hand to prove that $\mathsf{Server}_i(\mathbf{1}) = \mathcal{SB}_i$ it doesn't suffice that $\mathbf{1}$ is the top element w.r.t. $\preceq_c$, since we have to prove that *any* behaviour possesses at least a server.

Proposition 2.14

$$\forall \sigma \in \mathcal{SB}.\ \ \sigma \preceq_c \mathbf{1}\ \ \&\ \ \mathbf{1} \preceq_s \sigma.$$

Proof. The first part of the conjunction easily follows by the fact that, for $k = \mathsf{rank}(\sigma)$ we have $\mathsf{Server}_k(\sigma) \neq \emptyset$ by Lemma 2.13(i), and by the fact that it straightforwardly holds $\mathsf{Server}_k(\mathbf{1}) = \mathcal{SB}_k$.

For the second part: we recall that $\rho \in \mathsf{Client}_i(\mathbf{1})$ if and only if $\rho \in \mathcal{SB}_i$, $i \geq \mathsf{rank}(\mathbf{1}) = 0$ and for all ρ' if $\rho \| \mathbf{1} \Longrightarrow \rho' \| \mathbf{1} \not\longrightarrow$ then $\rho' = \mathbf{1}$. Therefore, in such a case, we get $\rho \in \mathsf{Client}_i(\mathbf{1})$ if and only if for all ρ' if $\rho \Longrightarrow \rho' \not\longrightarrow$ then $\rho' = \mathbf{1}$ (trivially $\mathbf{1}$ itself satisfies the condition). But any such ρ belongs to $\mathsf{Client}_k(\sigma)$ as soon as $k \geq \max(\mathsf{rank}(\rho), \mathsf{rank}(\sigma))$; in particular this is the case for $\rho = \sigma$: hence $\emptyset \neq \mathsf{Client}_k(\mathbf{1}) \subseteq \mathsf{Client}_k(\sigma)$, i.e. $\mathbf{1} \preceq^k_s \sigma$ if $k \geq \mathsf{rank}(\sigma)$, and hence $\mathbf{1} \preceq_s \sigma$ by definition. □

2.2 A non-stratified characterization of the sub-behaviour relations.

Even if the definition of the sub-behaviour relations relies on the stratification of $\mathcal{SB}$, they can be given a simple and natural characterization in terms of the subset relation between sets of clients, servers and peers, defined as the unions of the sets $\mathsf{Client}_i(\sigma)$, $\mathsf{Server}_i(\rho)$ and $\mathsf{Peer}_i(\sigma)$ respectively.

Definition 2.15 ($\mathsf{Client}(\sigma)$, $\mathsf{Server}(\rho)$ and $\mathsf{Peer}(\sigma)$)
For $\sigma, \rho \in \mathcal{SB}$, we define the sets:

$$\mathsf{Client}(\sigma) = \bigcup_{i\in\mathbb{N}} \mathsf{Client}_i(\sigma) \qquad \mathsf{Server}(\rho) = \bigcup_{i\in\mathbb{N}} \mathsf{Server}_i(\rho)$$
$$\mathsf{Peer}(\sigma) = \bigcup_{i\in\mathbb{N}} \mathsf{Peer}_i(\sigma)$$

From Lemma 2.13 we know that $\mathsf{Client}(\sigma)$, $\mathsf{Server}(\sigma)$ and $\mathsf{Peer}(\sigma)$ are non trivial notions for arbitrary σ.

Corollary 2.16 $\quad \forall \sigma \in \mathcal{SB}.\ \mathsf{Client}(\sigma) \neq \emptyset \ \ \&\ \ \mathsf{Server}(\sigma) \neq \emptyset \ \ \&\ \ \mathsf{Peer}(\sigma) \neq \emptyset.$

The characterizations we aim at are then as follows:

$$\begin{gathered} \sigma \preceq_s \sigma' \Leftrightarrow \mathsf{Client}(\sigma) \subseteq \mathsf{Client}(\sigma') \qquad \rho \preceq_c \rho' \Leftrightarrow \mathsf{Server}(\rho) \subseteq \mathsf{Server}(\rho') \\ \sigma \preceq_* \sigma' \Leftrightarrow \mathsf{Peer}(\sigma) \subseteq \mathsf{Peer}(\sigma') \end{gathered} \tag{6}$$

Moreover, we expect that for $\mathsf{Peer}(\sigma)$ and $\preceq_*$ the following hold:

$$\mathsf{Peer}(\sigma) = \mathsf{Client}(\sigma) \cap \mathsf{Server}(\sigma) \tag{7}$$

and

$$\preceq_* = \preceq_c \cap \preceq_s \tag{8}$$

If (6) is the case then $\preceq_s$, $\preceq_c$ and $\preceq_*$ are all preorders. However, there are some difficulties in proving this. Suppose in fact that $\sigma \preceq_s \sigma'$. Then we know that the set of clients of σ is non empty and included into that of σ', but this is the case only below a certain rank. As a matter of fact, if we know that $\mathsf{Client}_i(\sigma) \subseteq \mathsf{Client}_i(\sigma')$, it is not obvious that a client ρ of σ, possibly of higher rank than i, will be a client of σ' as well. Similar remarks concerning stratification levels can be made about the other relations and about the claim (7). For what concerns Claim (8), and in particular the $\subseteq$ direction, the problem depends on the two separate quantifications over clients and servers of a behaviour that are not equivalent to a singular quantification over their peers. Claim (8) will be proved in Section 3 by means of the coinductive characterisation of our sub-behaviour relations. We devote instead the remaining part of this section to the proofs of claims (6) and (7).

We begin the study by stating the following simple fact, easily descending by the corresponing definitions.

Fact 2.17 *For any $i \leq j$:*

$$\mathsf{Client}_i(\sigma) \subseteq \mathsf{Client}_j(\sigma) \quad \& \quad \mathsf{Server}_i(\sigma) \subseteq \mathsf{Server}_j(\sigma) \quad \& \quad \mathsf{Peer}_i(\sigma) \subseteq \mathsf{Peer}_j(\sigma).$$

From the above fact, the following Lemma easily descends.

Lemma 2.18

i) $\rho \in \mathsf{Client}(\sigma)$ & $k = \max(\mathsf{rank}(\rho), \mathsf{rank}(\sigma)) \quad \Rightarrow \quad \rho \in \mathsf{Client}_k(\sigma)$;

ii) $\sigma \in \mathsf{Server}(\rho)$ & $k = \max(\mathsf{rank}(\rho), \mathsf{rank}(\sigma)) \quad \Rightarrow \quad \sigma \in \mathsf{Server}_k(\rho)$;

iii) $\sigma \in \mathsf{Peer}(\rho)$ & $k = \max(\mathsf{rank}(\rho), \mathsf{rank}(\sigma)) \quad \Rightarrow \quad \sigma \in \mathsf{Peer}_k(\rho)$.

Now we are able to prove Claim (7).

Proposition 2.19

$$\mathsf{Peer}(\sigma) = \mathsf{Client}(\sigma) \cap \mathsf{Server}(\sigma)$$

Proof. ($\subseteq$) Immedate, by definition of $\mathsf{Peer}(\sigma)$, $\mathsf{Client}(\sigma)$ and $\mathsf{Server}(\sigma)$.
($\supseteq$) Let $\rho \in \mathsf{Client}(\sigma) \cap \mathsf{Server}(\sigma)$ and let $k = \max(\mathsf{rank}(\rho), \mathsf{rank}(\sigma))$. By Lemma 2.18 we have that $\rho \in \mathsf{Client}_k(\sigma) \cap \mathsf{Server}_k(\rho)$, and hence $\rho \in \mathsf{Peer}(\sigma)$ by definition.

We now move towards the proof of the right-to-left implications of claims (6).

Lemma 2.20

i) $\mathsf{Client}(\sigma) \subseteq \mathsf{Client}(\sigma') \quad \Rightarrow \quad \exists i \in \mathbb{N}.\ \sigma \preceq_s^i \sigma'$;

ii) $\mathsf{Server}(\rho) \subseteq \mathsf{Server}(\rho') \quad \Rightarrow \quad \exists i \in \mathbb{N}.\ \rho \preceq_c^i \rho'$;

iii) $\mathsf{Peer}(\sigma) \subseteq \mathsf{Peer}(\sigma') \quad \Rightarrow \quad \exists i \in \mathbb{N}.\ \sigma \preceq_*^i \sigma'$.

Proof. (i) Let $k = \max(\mathsf{rank}(\sigma), \mathsf{rank}(\sigma'))$. Then $\overline{\sigma} \in \mathsf{Client}_k(\sigma) \neq \emptyset$ by Lemma 2.13 (i) and Fact 2.17. On the other hand, if $\rho \in \mathsf{Client}_k(\sigma) \subseteq \mathsf{Client}(\sigma) \subseteq \mathsf{Client}(\sigma')$ we have that $\mathsf{rank}(\rho) \leq k$ and, by Lemma 2.18, that $\rho \in \mathsf{Client}_h(\sigma')$ where $h = \max(\mathsf{rank}(\rho), \mathsf{rank}(\sigma'))$. It follows that $h \leq k$ and hence $\rho \in \mathsf{Client}_k(\sigma')$ by Fact 2.17.
(ii), (iii). Similar to (i). □
The above lemma implies that:

$$\mathsf{Client}(\sigma) \subseteq \mathsf{Client}(\sigma') \Rightarrow \sigma \preceq_s \sigma' \qquad \mathsf{Server}(\rho) \subseteq \mathsf{Server}(\rho') \Rightarrow \rho \preceq_c \rho'$$
$$\mathsf{Client}(\sigma) \subseteq \mathsf{Client}(\sigma') \Rightarrow \sigma \preceq_s \sigma'$$

Establishing the left-to-right implications of (6) is more involved, and a precise definition of the concept of synchronisation is in order.

Definition 2.21 *Given* $\alpha, \beta \in \mathbf{Act}$ *and* $k \in \mathbb{N}$ *we say that* α and β synchronise below k *and write* α synch β below k, *if one of the following cases occur:*

i) $\alpha, \beta \in \mathbf{Act}_0$ *and* $\alpha = \overline{\beta}$ *and* k *is arbitrary (even* 0*);*

ii) $\alpha = !{[\sigma^p]}$ *and* $\beta = ?(\tau^p)$ *or* $\alpha = ?(\tau^p)$ *and* $\beta = !{[\sigma^p]}$ *and* $\sigma \preceq_p^h \tau$ *for some* $h < k$.

We then say that α and β synchronise, *written* α synch β, *if* α synch β *below* k *for some* k.

Observe that if α synch β and k bounds above both $\mathsf{rank}(\alpha)$ and $\mathsf{rank}(\beta)$ then α synch β below k.

Let $s = \alpha_1 \cdots \alpha_m$ and $t = \beta_1 \cdots \beta_n$ be sequences in $\mathbf{Act}^*$: then we say that s *and* t *synchronise pointwise* (or just *synchronise*), and write s synch t, if $m = n$ and α_i synch β_i for all $i = 1, \ldots, m$. We also write $|s|$ for the length of $s \in \mathbf{Act}^*$.

Lemma 2.22 *Let* $\rho, \sigma \in \mathcal{SB}$:

i) $\rho \| \sigma \Longrightarrow \rho' \| \sigma'$ *for some* $\rho', \sigma' \in \mathcal{SB}$ *if and only if there exist* $s, t \in \mathbf{Act}^*$ *such that* s synch t *and* $\rho \stackrel{s}{\Longrightarrow} \rho'$ *and* $\sigma \stackrel{t}{\Longrightarrow} \sigma'$;

ii) $\rho \| \sigma \not\longrightarrow$ *if and only if either* $\rho = \mathbf{1}$ *or* $\sigma = \mathbf{1}$ *or*

$$\rho \not\longrightarrow \;\&\; \sigma \not\longrightarrow \;\&\; \neg \exists \alpha, \beta \in \mathbf{Act}.\, [\alpha \;\mathsf{synch}\; \beta \;\&\; \rho \stackrel{\alpha}{\longrightarrow} \;\&\; \sigma \stackrel{\beta}{\longrightarrow}] \;;$$

iii) if $\rho \dashv \sigma$ *and* $\rho \| \sigma \Longrightarrow \rho' \| \sigma'$ *then* $\rho' \dashv \sigma'$;

iv) if $\rho \perp \sigma$ *and* $\rho \| \sigma \Longrightarrow \rho' \| \sigma'$ *then* $\rho' \perp \sigma'$

Proof. Easy consequences of Def. 2.8. □

Lemma 2.23 *Let* $\sigma, \sigma', \rho, \rho' \in \mathcal{SB}$. *For all* $k \in \mathbb{N}$:

i) if $i \leq k$ *then* $\sigma \preceq_s^i \sigma' \;\Rightarrow\; \sigma \preceq_s^k \sigma'$;

ii) if $i \leq k$ *then* $\rho \preceq_c^i \rho' \;\Rightarrow\; \rho \preceq_c^k \rho'$;

iii) if $i \leq k$ *then* $\rho \preceq_*^i \rho' \;\Rightarrow\; \rho \preceq_*^k \rho'$.

Proof. We prove (i), (ii) and (iii) by simultaneous induction over k. The case $k = 0$ is trivial since then $i = k$. Let $k > 0$; from $\sigma \preceq_s^i \sigma'$ and $i \leq k$ it follows that $\emptyset \neq \mathsf{Client}_i(\sigma) \subseteq \mathsf{Client}_k(\sigma)$, i.e. $\mathsf{Client}_k(\sigma) \neq \emptyset$. To prove that $\mathsf{Client}_k(\sigma) \subseteq \mathsf{Client}_k(\sigma')$ we reason by contradiction: suppose that there exists some $\rho \in \mathsf{Client}_k(\sigma) \setminus \mathsf{Client}_k(\sigma')$. By Lemma 2.22 there exist $s \in \mathbf{Act}^*$ of minimal length, $\rho_0 \in \mathcal{SB}$ and $\alpha \in \mathbf{Act}$ such that $\rho \stackrel{s}{\Longrightarrow} \rho_0 \stackrel{\alpha}{\Longrightarrow}$ and $\sigma' \stackrel{\tilde{s}}{\Longrightarrow} \sigma'_0$ with $\tilde{s}$ pointwise synchronising with s, but for any β synchronising with α, $\sigma'_0 \stackrel{\beta}{\not\Longrightarrow}$. For simplicity, let us assume that $|s| = |\tilde{s}| = 0$, that is $\rho_0 = \rho$ and $\sigma'_0 = \sigma'$: indeed the general case can be treated by iterating the following argument. By assumption $\rho \dashv \sigma$, hence also $\alpha.\mathbf{1} \dashv \sigma$, which implies that $\sigma \stackrel{\gamma}{\Longrightarrow}$ for some γ s.t. α synch γ. Since $\mathsf{rank}(\alpha) \leq \mathsf{rank}(\rho) \leq k$ and $\mathsf{rank}(\gamma) \leq \mathsf{rank}(\sigma) \leq k$ we know that α synch γ below k.

On the other hand, since by definition $\overline{\gamma}.\mathbf{1} \stackrel{\overline{\gamma}}{\longrightarrow} \mathbf{1}$ and since we showed that $\sigma \stackrel{\gamma}{\Longrightarrow}$, we have that, by definition of $\dashv$, that $\overline{\gamma}.\mathbf{1} \dashv \sigma$. Then, since $h = \mathsf{rank}(\overline{\gamma}.\mathbf{1}) = \mathsf{rank}(\gamma) \leq \mathsf{rank}(\sigma) \leq i$, we know that $\overline{\gamma}.\mathbf{1} \in \mathsf{Client}_i(\sigma) \subseteq \mathsf{Client}_i(\sigma')$ by the hypothesis that $\sigma \preceq_s^i \sigma'$. It follows that $\sigma' \stackrel{\delta}{\Longrightarrow}$ for some δ s.t. $\overline{\gamma}$ synch δ, and again this is the case below k.

Now let us consider the possible cases according to the shape of α, and consequently of γ and δ:

$\alpha \in \mathbf{Act}_0$: then $\mathsf{rank}(\alpha) = \mathsf{rank}(\gamma) = \mathsf{rank}(\delta) = 0$, hence $\alpha, \gamma, \delta \in \mathbf{Act}_0$ so that there exists $a \in \mathcal{N}$ s.t. either $\alpha = \overline{\gamma} = a$ and $\gamma = \delta = \overline{a}$ or $\alpha = \overline{\gamma} = \overline{a}$ and $\gamma = \delta = a$: in both cases α synch δ.

$\alpha = !{[\tau_0^p]}$: then by the fact that α synch γ below k we have $\gamma = ?(\tau_1^p)$ for some τ_1 s.t. $\tau_0 \preceq_p^{k_0} \tau_1$ where $k_0 < k$. It follows that $\overline{\gamma} = !{[\tau_1^p]}$, and by $\overline{\gamma}$ synch δ we deduce that $\delta = ?(\tau_2^p)$ for some τ_2 s.t. $\tau_1 \preceq_p^{k_1} \tau_2$ where also $k_1 < k$. By induction hypothesis (i),(ii) or (iii)according to p, we have that $\tau_0 \preceq_p^h \tau_1 \preceq_p^h \tau_2$ where $h = \max(k_0, k_1) < k$ i.e. $\tau_0 \preceq_p^h \tau_2$ and we conclude that α synch δ.

$\alpha = ?(\tau_0^p)$: in this case we have $\alpha = ?(\tau_0^p)$, $\gamma = !{[\tau_1^p]}$ and $\delta = !{[\tau_2^p]}$ where $\tau_1 \preceq_p^{k_0} \tau_0$ and $\tau_2 \preceq_p^{k_1} \tau_1$, where $k_0, k_1 < k$; then we argue as in the previous case concluding that α synch δ.

In all possible cases we get a contradiction w.r.t. the assertion that $\sigma' \stackrel{\beta}{\not\Longrightarrow}$ for any β synchronising with α. The proofs of (ii) and (iii) are similar and we are done. □

Theorem 2.24 *For all $\sigma, \sigma', \rho, \rho' \in \mathcal{SB}$:*

$$\sigma \preceq_s \sigma' \Leftrightarrow \mathsf{Client}(\sigma) \subseteq \mathsf{Client}(\sigma') \qquad \rho \preceq_c \rho' \Leftrightarrow \mathsf{Server}(\rho) \subseteq \mathsf{Server}(\rho')$$
$$\sigma \preceq_* \sigma' \Leftrightarrow \mathsf{Peer}(\sigma) \subseteq \mathsf{Peer}(\sigma')$$

Proof. The $\Leftarrow$ implications follow by Lemma 2.20. To show $\Rightarrow$ suppose that $\sigma \preceq_s \sigma'$, namely $\sigma \preceq_s^i \sigma'$ for some i; if $\rho \in \mathsf{Client}(\sigma)$ then $\rho \in \mathsf{Client}_j(\sigma)$ for some j; then for $k = \max(i, j)$ we have:

$$\rho \in \mathsf{Client}_j(\sigma) \subseteq \mathsf{Client}_k(\sigma) \subseteq \mathsf{Client}_k(\sigma') \subseteq \mathsf{Client}(\sigma'),$$

by Fact 2.17 and (i) of Lemma 2.23. This proves $\sigma \preceq_s \sigma' \Rightarrow \mathsf{Client}(\sigma) \subseteq \mathsf{Client}(\sigma')$; the other implications follow in a similar way using Fact 2.17 and (ii),(iii) of Lemma 2.23.

Corollary 2.25 *$\preceq_s$, $\preceq_c$ and $\preceq_*$ are preorders, that is reflexive and transitive relations.*

Proof. Immediate by Theorem 2.24. □

We end up with a nice property of syntactical duality w.r.t. the preorders $\preceq_c$, $\preceq_s$ and$\preceq_*$. Let us first establish a lemma.

Lemma 2.26 *For all $\rho, \sigma, \tau \in \mathcal{SB}$,*

i) If $\rho \dashv \tau$ and $\overline{\tau} \dashv \sigma$ then $\rho \dashv \sigma$;

ii) If $\rho \perp \tau$ and $\overline{\tau} \perp \sigma$ then $\rho \perp \sigma$.

Proof. (i) It suffices to prove that if $\rho \stackrel{\alpha}{\Longrightarrow}$ then there exists γ s.t. $\sigma \stackrel{\gamma}{\Longrightarrow}$ and α synch γ. If $\rho \stackrel{\alpha}{\Longrightarrow}$ then, by the hypothesis $\rho \dashv \tau$, there exists β s.t. $\tau \stackrel{\beta}{\Longrightarrow}$ and

α synch β. On the other hand if $\tau \stackrel{\beta}{\Longrightarrow}$ then $\overline{\tau} \stackrel{\overline{\beta}}{\Longrightarrow}$, so that, by the assumption $\overline{\tau} \dashv \sigma$, we have that $\sigma \stackrel{\gamma}{\Longrightarrow}$ for some γ s.t. $\overline{\beta}$ synch γ. We now check that α synch γ by cases of α.

Let $\alpha \in \mathbf{Act}_0$: then either $\alpha = a$ or $\alpha = \overline{a}$ for some $a \in \mathcal{N}$. In the first case we have that $\beta = \overline{a}$ so that $\overline{\beta} = \overline{\overline{a}} = a$ and we obtain that $\gamma = \overline{a}$. If instead $\alpha = \overline{a}$, we obtain by the same reasoning that $\gamma = a$, and in both cases α synch γ.

If $\alpha = ![\rho_0^p]$ then $\beta = ?(\tau_0^p)$ for some τ_0 s.t. $\rho_0 \preceq_p \tau_0$. It follows that $\overline{\beta} = ![\tau_0^p]$; therefore $\gamma = ?(\sigma_0^p)$ for some σ_0 s.t. $\tau_0 \preceq_p \sigma_0$. By Corollary 2.25 we know that $\preceq_p$ is transitive for both $p = c, s$, hence we deduce that $\rho_0 \preceq_p \sigma_0$ and we conclude that α synch γ.

The case $\alpha = ?(\rho_0^p)$ is treated similarly, by deducing that $\gamma = ![\sigma_0^p]$, this time with $\sigma_0 \preceq_p \tau_0 \preceq_p \rho_0$ for some τ_0 s.t. $\beta = ![\tau_0^p]$.
(ii) Easy by definition fo $\perp$ and point (i), observing that the $\bar{\cdot}$ operation is involutive. □

Proposition 2.27 *Let $\tau \in \mathcal{SB}$. Then*

i) $\overline{\tau}$ is the minimum client of τ, i.e. $\forall \rho \in \mathsf{Client}(\tau).\ \overline{\tau} \preceq_c \rho$;

ii) $\overline{\tau}$ is the minimum server of τ, i.e. $\forall \sigma \in \mathsf{Server}(\tau).\ \overline{\tau} \preceq_s \sigma$;

iii) $\overline{\tau}$ is the minimum peer of τ, i.e. $\forall \rho \in \mathsf{Peer}(\tau).\ \overline{\tau} \preceq_* \rho$.

Proof. We observe that, by Lemma 2.13 and Definition 2.15, $\overline{\tau} \in \mathsf{Client}(\tau) \cap \mathsf{Server}(\tau)$. Hence it remains to show the minimality property w.r.t. $\preceq_c$ and $\preceq_s$, respectively.
(i) Let $\rho \in \mathsf{Client}(\tau)$. In order to establish $\overline{\tau} \preceq_c \rho$, let $\sigma \in \mathsf{Server}(\overline{\tau})$. Then we have $\rho \dashv \tau$ and $\overline{\tau} \dashv \sigma$. By Lemma 2.26 we know that $\rho \dashv \sigma$, i.e. $\sigma \in \mathsf{Server}(\rho)$. Hence $\overline{\tau} \preceq_c \rho$. The proofs of (ii) and (iii) are similar, using Lemma 2.26 and observing that the $\bar{\cdot}$ operation is involutive. □

Remark 2.28 *The properties in Proposition 2.27 depend on the lack of* implicit *nondeterminism in session behaviours (reflecting the same characteristic of session types). The nondeterminism in our system is in fact only* explicit*, since it is due exclusively to the $\oplus$ operator.*

A light form of implicit nondeterminism could be introduced by relaxing the constraint imposing prefixes in a sum $+$ or $\oplus$ to be pairwise distinct: in fact, if we let $a + a.b$ to be a session behaviour, $(a + a.b) \| \overline{a}$ reduces both to $\mathbf{1}$ and to b. Then $\overline{a} \oplus \overline{a}.\overline{b} \not\dashv a + a.b$ and the minimum of $\mathsf{Client}(a + a.b)$ is actually $\overline{a}$. On the other hand, $a + a.b \not\dashv \overline{a} \oplus \overline{a}.\overline{b}$ and the minimum of $\mathsf{Server}(a + a.b)$ is $\overline{a}.\overline{b}$. But some behaviours could have no minimum client or server at all, e.g. $\mathsf{Server}(a.\overline{b} + a.\overline{c}) = \emptyset$.

Relaxing the constraint that $a_1, \ldots, a_n \in \mathcal{N}$ in $a_1.\sigma_1 + \cdots + a_n$ and $\overline{a}_1, \ldots, \overline{a}_n \in \overline{\mathcal{N}}$ in $\overline{a}_1.\sigma_1 \oplus \cdots \oplus \overline{a}_n.\sigma_n$, would, instead, introduce even more nondeterminisms, taking us completely outside of the "session" context.

3 Coinductive characterizations

The definitions of compliance and sub-behaviour relations in the previous section remain unmanageable because of their loose connection with the algebraic

structure of behaviours. To remedy the deficiency, this session is devoted to the coinductive characterisation of these concepts.

We start from some remarks. The syntax of session behaviours prevents infinite $\longrightarrow$ reductions. A raw behavioural expression like $\mathsf{rec}\, x.x$ is not, in fact, a session behaviour. Also an expression like $\mathsf{rec}\, x.(y \oplus z)$ is not a proper session behaviour (not even a raw expression), since we allow the $+$ and $\oplus$ operators only with *prefixed* expressions. Such syntactical restrictions, however, do not prevent an abstract representation of branching and selection types. At the same time they rule out diverging expressions like $\mathsf{rec}\, x.x$ and $\mathsf{rec}\, x.\mathsf{rec}\, y.(x \oplus x)$ which would be meaningless in the present setting. This is made precise by the following lemma.

Lemma 3.1 *For any $\sigma \in \mathcal{SB}$, there exists no infinite $\longrightarrow$ reduction out of it. Moreover, given $\sigma \in \mathcal{SB}$, there exists a unique finite and non empty $R \subseteq \mathcal{SB}$ such that $R = \{\sigma' \in \mathcal{SB} \mid \sigma \Longrightarrow \sigma' \not\longrightarrow\}$, which takes one of the following forms:*

$$\{\mathbf{1}\},\quad \{\sum_{i=1}^{n} a_i.\sigma_i\},\quad \{\overline{a}_1.\sigma_1,\ldots,\overline{a}_n.\sigma_n\},\quad \{?(\sigma_1^p)\sigma_2\},\quad \{![\sigma_1^p]\sigma_2\}\quad (n>0).$$

Proof. By cases of $\sigma \in \mathcal{SB}$: in particular, if $\sigma = \mathsf{rec}\, x.\sigma'$ we know that $\sigma' \neq x$ and that, in case σ' be a $\oplus$-term, its components are prefixed. So the longest $\longrightarrow$ reduction sequence out of σ necessarily consists in a number of $\xrightarrow{\mathsf{rec}}$ steps (which is equal to the number of occurrences of rec prefixing σ, since σ can actually be of the form $\mathsf{rec}\, x.\mathsf{rec}\, x_2.\ldots.\mathsf{rec}\, x_n.\sigma''$), followed by at most one $\xrightarrow{\oplus}$ step. From the above reasoning it easily follows also the second part of the statement. □

Definition 3.2 *For any $\sigma \in \mathcal{SB}$ and $R \subseteq \mathcal{SB}$, we define*

$$\sigma \Downarrow R \quad \textit{if and only if} \quad R = \{\sigma' \in \mathcal{SB} \mid \sigma \Longrightarrow \sigma' \not\longrightarrow\}$$

As a shorthand, we shall write $\sigma \Downarrow \mathbf{1}$, $\sigma \Downarrow \sum_{i=1}^{n} a_i.\sigma_i$, $\sigma \Downarrow \bigoplus_{i=1}^{n} \overline{a}_i.\sigma_i$, $\sigma \Downarrow ?(\sigma_1^p)\sigma_2$ and $\sigma \Downarrow ![\sigma_1^p]\sigma_2$ whenever $\sigma \Downarrow R$ and R has one of the four forms in Lemma 3.1, respectively. Equivalently $\sigma \Downarrow \tau$ if and only if τ is the unique session behaviour such that $\sigma \xrightarrow{\mathsf{rec}}{}^{*} \tau$. By this we have:

Corollary 3.3 *For any $\sigma, \tau \in \mathcal{SB}$, if $\sigma \Downarrow \tau$ then $\mathsf{rank}(\sigma) = \mathsf{rank}(\tau)$.*

Proof. By the above remark and Lemma 2.6. □

Definition 3.4 (Coinductive Client/Server/Peer Relation Triple) *i) The operator $\mathcal{H} : (\mathcal{P}(\mathcal{SB} \times \mathcal{SB} \times \mathcal{SB}))^2 \to (\mathcal{P}(\mathcal{SB} \times \mathcal{SB} \times \mathcal{SB}))^2$ is defined as follows: for any triple of relations $(\mathcal{R}_c, \mathcal{R}_s, \mathcal{R}_*) \subseteq \mathcal{SB}^2 \times \mathcal{SB}^2 \times \mathcal{SB}^2$, $((\sigma_c, \tau_c), (\sigma_s, \tau_s), (\sigma_*, \tau_*)) \in \mathcal{H}(\mathcal{R}_c, \mathcal{R}_s, \mathcal{R}_*)$ if and only if:*
either $\tau_c \Downarrow \mathbf{1}$ & $\sigma_s \Downarrow \mathbf{1}$ & $\tau_* \Downarrow \mathbf{1}$ & $\sigma_* \Downarrow \mathbf{1}$,
or the following conditions hold, where $p, q \in \{s, c, *\}$:

a) $\sigma_p \Downarrow \sum_{i\in I} a_i.\sigma_{p_i} \;\Rightarrow\; \exists J \supseteq I.\ \tau_p \Downarrow \sum_{j\in J} a_j.\tau_{p_j}$ & $\forall i \in I.\ \sigma_{p_i} \mathcal{R}_p \tau_{p_i}$

b) $\sigma_p \Downarrow \bigoplus_{i\in I} \overline{a}_i.\sigma_{p_i} \;\Rightarrow\; \exists J \subseteq I.\ \tau_p \Downarrow \bigoplus_{j\in J} \overline{a}_j.\tau_{p_j} \;\&\; \forall j \in J.\ \sigma_{p_j} \mathcal{R}_p \tau_{p_j}$

c) $\sigma_p \Downarrow ?(\sigma_1^q)\sigma_2 \;\Rightarrow\; \tau_p \Downarrow ?(\tau_1^q)\tau_2 \;\&\; \sigma_1 \mathcal{R}_q \tau_1 \;\&\; \sigma_2 \mathcal{R}_p \tau_2$

d) $\sigma_p \Downarrow ![\sigma_1^q]\sigma_2 \;\Rightarrow\; \tau \Downarrow ![\tau_1^q]\tau_2 \;\&\; \tau_1 \mathcal{R}_q \sigma_1 \;\&\; \sigma_2 \mathcal{R}_p \tau_2$

ii) A triple of relations $(\mathcal{R}_c, \mathcal{R}_s, \mathcal{R}_*) \subseteq \mathcal{SB}^2 \times \mathcal{SB}^2 \times \mathcal{SB}^2$ *is a* coinductive client/server/peer relation triple *if and only if* $(\mathcal{R}_c, \mathcal{R}_s, \mathcal{R}_*) \subseteq \mathcal{H}(\mathcal{R}_c, \mathcal{R}_s, \mathcal{R}_*)$.

Since $\mathcal{R}_c$, $\mathcal{R}_s$ and $\mathcal{R}_*$ occur covariantly in the clauses defining $\mathcal{H}(\mathcal{R}_c, \mathcal{R}_s, \mathcal{R}_*)$, the operator $\mathcal{H}$ is monotonic. Then the following fact immediately follows by Tarsky theorem:

Fact 3.5 *Let* $\mathcal{H}^0 = \mathcal{SB}^2 \times \mathcal{SB}^2 \times \mathcal{SB}^2$ *and* $\mathcal{H}^{k+1} = \mathcal{H}(\mathcal{H}^k)$*; then*

$$\nu(\mathcal{H}) \;=\; \bigcup\{(\mathcal{R}_c, \mathcal{R}_s, \mathcal{R}_*) \subseteq \mathcal{SB}^2 \times \mathcal{SB}^2 \times \mathcal{SB}^2 \mid (\mathcal{R}_c, \mathcal{R}_s, \mathcal{R}_*) \subseteq \mathcal{H}(\mathcal{R}_c, \mathcal{R}_s, \mathcal{R}_*)\} \;=\; \bigcap_{k\in\mathbb{N}} \mathcal{H}^k$$

is the greatest fixed point of $\mathcal{H}$.

Then we define coinductively the following relations:

Definition 3.6

$$(\preceq_c^{co.k}, \preceq_s^{co.k}, \preceq_*^{co.k}) =_{def} \mathcal{H}^k \qquad \textit{and} \qquad (\preceq_c^{co}, \preceq_s^{co}, \preceq_*^{co}) =_{def} \nu(\mathcal{H}),$$

where $\mathcal{H}^k$ *is defined as in Fact 3.5.*

Lemma 3.7 $(\preceq_c, \preceq_s, \preceq_*)$ *is a client/server/peer relation triple.*

Proof. It suffices to check the clauses of Definition 3.4 with $\preceq_c$, $\preceq_s$ and $\preceq_*$ in place of $\mathcal{R}_c$, $\mathcal{R}_s$ and $\mathcal{R}_*$ respectively. In case $\sigma \preceq_c \tau$, $\sigma' \preceq_s \tau'$ and $\sigma'' \preceq_* \tau''$ with $\tau \Downarrow \mathbf{1}$, $\sigma' \Downarrow \mathbf{1}$, $\tau'' \Downarrow \mathbf{1}$, and $\sigma \Downarrow \mathbf{1}$ the first clause (before (a)) of the definition is satisfied and we are done. We go on now treating the case of $\preceq_s$ only, since $\preceq_c$ and $\preceq_*$ can be treated in a similar way.

Suppose that $\sigma \Downarrow \sum_{i\in I} a_i.\sigma_i$: then we observe that $\rho \dashv \sigma$ and $\rho \not\Downarrow \mathbf{1}$ if and only if $\rho = \bigoplus_{h\in H} \overline{a}_h.\rho_h$ for some $H \subseteq I$ and $\rho_h \dashv \sigma_h$ for all $h \in H$. Moreover, by assumption, $\rho \in \mathsf{Client}(\tau)$ which implies that $\tau \Downarrow \sum_{j\in J} a_j.\tau_j$ for some $J \supseteq H$ such that $\rho_h \dashv \tau_h$ for all $h \in H$. We can now show that for all $i \in I$ $\mathsf{Client}(\sigma_i) \subseteq \mathsf{Client}(\tau_i)$ and hence, by Theorem 2.24, $\sigma_i \preceq_s \tau_i$. In fact, let $\{\rho_i\}_{i\in I}$ be any set such that $\rho_i \in \mathsf{Client}(\sigma_i)$. We observe that, for $\rho = \bigoplus_{i\in I} \overline{a}_i.\rho_i$, $\rho \dashv \sigma$ and hence $\rho_i \in \mathsf{Client}(\tau_i)$ for all $i \in I$ by the preceding observations.

If $\sigma \Downarrow \bigoplus_{i\in I} \overline{a}_i.\sigma_i$ the proof is as before. Let us now consider the case $\sigma \Downarrow ?(\sigma_1^q)\sigma_2$. Then $\rho \dashv \sigma$ and $\rho \not\Downarrow \mathbf{1}$ if and only if $\rho = ![\rho_1^q]\rho_2$ such that $\rho_1 \preceq_q \sigma_1$ and $\rho_2 \dashv \sigma_2$. Now the hypothesis $\sigma \preceq_s \tau$ implies $\tau \Downarrow ?(\tau_1^q)\tau_2$ where it has to be the case that $\rho_1 \preceq_q \tau_1$ and $\rho_2 \dashv \tau_2$. By the arbitrary choice of ρ we can take $\rho_1 = \sigma_1$, so that in particular we have $\sigma_1 \preceq_q \tau_1$ and, by letting ρ_2 vary over the whole $\mathsf{Client}(\sigma_2)$, we can conclude that $\mathsf{Client}(\sigma_2) \subseteq \mathsf{Client}(\tau_2)$, that is $\sigma_2 \preceq_s \tau_2$ as desired.

The case $\sigma \Downarrow ![\sigma_1^q]\sigma_2$ is similar to the last one, and we are done. □

Lemma 3.8 *For any $\sigma, \tau \in \mathcal{SB}$ and $p = s, c, *$:*

$$\sigma \preceq_p^{co} \tau \quad \Rightarrow \quad \sigma \preceq_p \tau.$$

Proof. We shall prove that $\sigma \preceq_p^{co} \tau$ implies $\sigma \preceq_p^k \tau$ for $k = \max(\mathsf{rank}(\sigma), \mathsf{rank}(\tau))$ (which suffices since $\preceq_p = \bigcup_i \preceq_p^i$) by double induction: primary induction over k and secondary induction over the length of certain reduction sequences. In particular, when applying the secondary induction, we shall show, equivalently, the contraposite implication. To this aim recall that $\sigma \not\preceq_p^k \tau$ if and only if $\mathsf{Client}_k(\sigma) \not\subseteq \mathsf{Client}_k(\tau)$ when $p = s$, $\mathsf{Server}_k(\sigma) \not\subseteq \mathsf{Server}_k(\tau)$ when $p = c$ and $\mathsf{Peer}_k(\sigma) \not\subseteq \mathsf{Peer}_k(\tau)$ when $p = *$ (in fact by the assumption that $k \geq \mathsf{rank}(\sigma)$ we know that $\overline{\sigma} \in \mathsf{Client}_k(\sigma) \cap \mathsf{Server}_k(\sigma) \cap \mathsf{Peer}_k(\sigma)$ and therefore $\mathsf{Client}_k(\sigma)$, $\mathsf{Server}_k(\tau)$ and $\mathsf{Peer}_k(\tau)$ are all nonempty). Pick some $\rho \in \mathsf{Client}_k(\sigma) \setminus \mathsf{Client}_k(\tau)$ ($\rho \in \mathsf{Server}_k(\sigma) \setminus \mathsf{Server}_k(\tau)$ and $\rho \in \mathsf{Peer}_k(\sigma) \setminus \mathsf{Peer}_k(\tau)$, respectively): then there exist $\rho^1, \ldots, \rho^n$ such that $\rho^0 \| \tau^0 \Longrightarrow \rho^1 \| \tau^1 \Longrightarrow \cdots \Longrightarrow \rho^n \| \tau^n$ with $\rho^n \neq \mathbf{1}$ but $\rho^n \| \tau^n \not\longrightarrow$, where $\rho^0 = \rho$ and $\tau^0 = \tau$. Since $\rho \dashv \sigma$ we have that $\rho^0 \| \sigma^0 \Longrightarrow \rho^1 \| \sigma^1 \Longrightarrow \cdots \Longrightarrow \rho^n \| \sigma^n$ where $\sigma^0 = \sigma$. Then we argue by (secondary) induction over n that $\sigma \not\preceq_s^{co} \tau$ (the proofs that $\mathsf{Server}_k(\sigma) \not\subseteq \mathsf{Server}_k(\tau)$ implies $\sigma \not\preceq_c^{co} \tau$ and that $\mathsf{Peer}_k(\sigma) \not\subseteq \mathsf{Peer}_k(\tau)$ implies $\sigma \not\preceq_*^{co} \tau$ can be carried on by means of similar arguments and hence it will be omitted).

If $n = 0$ then $\rho^0 \| \tau^0 \not\longrightarrow$ (possibly disregarding a finite amount of internal reductions of ρ^0 and τ^0), namely $\rho \| \tau \not\longrightarrow$. From $\rho \dashv \sigma$ it follows that no clause in Definition 3.4 can be satisfied and we conclude that $\sigma \not\preceq_c^{co} \tau$ directly.

If $n > 0$ then suppose that $\sigma \Downarrow \sigma'$ and consider the possible shapes of σ' (which cannot be $\mathbf{1}$):

- $\sigma' = \sum_{i \in I} a_i.\sigma_i$: since $\rho = \rho^0 \dashv \sigma$ we deduce that $\rho^0 \Downarrow \bigoplus_{h \in H} \overline{a}_h.\rho_h$ and that $H \subseteq I$ and $\rho_h \dashv \sigma_h$ for all $h \in H$. In particular we have that $\rho^1 = \rho_i$ for some $i \in I$ such that $\sigma^1 = \sigma_i$. If not $\tau \Downarrow \sum_{j \in J} a_j.\tau_j$ or $J \not\supseteq I$ then we know that $\sigma \not\preceq_c^{co} \tau$ immediately; otherwise $J \supseteq I$, $\tau^1 = \tau_i$ and $\rho_i = \rho^1 \not\dashv \tau^1 = \tau_i$, which is witnessed by the reduction $\rho^1 \| \tau^1 \Longrightarrow \cdots \Longrightarrow \rho^n \| \tau^n$ of length $n - 1$. Then by the secondary induction hypothesis we know that $\sigma_i = \sigma^1 \not\preceq_s^{co} \tau^1 = \tau_i$, and we conclude that $\sigma \not\preceq_c^{co} \tau$.

- $\sigma' = \bigoplus_{i \in I} \overline{a}_i.\sigma_i$: symmetrically to the previous case we deduce that $\rho^0 \Downarrow \sum_{h \in H} a_h.\rho_h$, with $H \supseteq I$, and $\rho_i \dashv \sigma_i$ for all $i \in I$. If not $\tau \Downarrow \bigoplus_{j \in J} \overline{a}_j.\tau_j$ or $J \not\subseteq I$ then we are done; otherwise $J \subseteq I$, and since $\rho \| \tau \Longrightarrow \rho^1 \| \tau^1$ we know that $\tau^1 = \tau_j$ and $\rho^1 = \rho_j$ for some $j \in J \subseteq I \subseteq H$, and we have that $\rho_j \not\dashv \tau_j$ which is proved by the reduction $\rho^1 \| \tau^1 \Longrightarrow \cdots \Longrightarrow \rho^n \| \tau^n$ of length $n-1$: then the thesis follows by the secondary induction hypothesis, namely $\sigma_j = \sigma^1 \not\preceq_s^{co} \tau^1 = \tau_j$.

- $\sigma' = ?(\sigma_1^q)\sigma_2$: reasoning as before we have that $\rho^0 \Downarrow ![\rho_1^q]\rho_2$ with $\rho_1 \preceq_q^{co} \sigma_1$ and $\rho_2 \dashv \sigma_2$. We observe that $h = \max(\mathsf{rank}(\rho_1), \mathsf{rank}(\sigma_1)) < k$ because $\rho^0 = \rho \in \mathsf{Client}_k(\sigma)$ so that $\mathsf{rank}(\rho_1) < \mathsf{rank}(![\rho_1^q]\rho_2)$ and $\mathsf{rank}(![\rho_1^q]\rho_2) = \mathsf{rank}(\rho)$ by Lemma 2.6; on the other hand $\mathsf{rank}(\sigma_1) < \mathsf{rank}(\sigma') = \mathsf{rank}(\sigma)$, the last equation holding again by Lemma 2.6. By the primary induction hypothesis we have $\rho_1 \preceq_q^h \sigma_1$ so that $\rho^1 = \rho_2$ and $\sigma^1 = \sigma_2$. If not $\tau \Downarrow ?(\tau_1^q)\tau_2$ or $\sigma_1 \not\preceq_q^{co} \tau_1$ we are done; otherwise $\sigma_1 \preceq_q^{co} \tau_1$ which by the primary induction implies $\sigma_1 \preceq_q^l \tau_1$ for $l = \max(\mathsf{rank}(\sigma_1), \mathsf{rank}(\tau_1)) < k$:

let $m = \max(h, l)$, then $m < k$ and $\rho_1 \preceq_q^m \sigma_1 \preceq_q^m \tau_1$ by Lemma 2.23 that is $\rho_1 \preceq_q^m \tau_1$. We conclude that $\tau^1 = \tau_2$ and that $\rho_2 \not\dashv \tau_2$ is witnessed by a reduction of length $n-1$; therefore $\sigma_2 \not\preceq_s^{co} \tau_2$ by the secondary induction, which implies $\sigma \not\preceq_s^{co} \tau$.

$\sigma' = ![\sigma_1^q]\sigma_2$: similar to the last case. □

Theorem 3.9 $(\preceq_c, \preceq_s, \preceq_*)$ *is the largest client/server/peer relation triple w.r.t. componentwise inclusion, namely*

$$(\preceq_c, \preceq_s, \preceq_*) \;=\; (\preceq_c^{co}, \preceq_s^{co}, \preceq_*^{co}).$$

Proof. $(\preceq_c, \preceq_s, \preceq_*) \subseteq (\preceq_c^{co}, \preceq_s^{co}, \preceq_*^{co}) = \nu(\mathcal{H})$ follows by Lemma 3.7 and the fact that $\nu(\mathcal{H})$ is the largest client/server relation pair. The opposite inclusion is just Lemma 3.8. □

We can now show Claim (8). i.e. that the natural relationship among $\preceq_*$, $\preceq_c$ and $\preceq_s$ does hold.

Theorem 3.10

$$\preceq_* \;=\; \preceq_c \cap \preceq_s$$

Proof. ($\supseteq$) Assume that $\sigma \preceq_c \tau$ and $\sigma \preceq_s \tau$. If $\rho \in \mathsf{Peer}(\sigma)$ then $\rho \in \mathsf{Client}(\sigma)$ and $\rho \in \mathsf{Server}(\sigma)$ by Lemma 2.19. Hence, by assumption, $\rho \in \mathsf{Client}(\tau) \cap \mathsf{Server}(\tau)$. Then, by Lemma 2.19 again, $\rho \in \mathsf{Peer}(\tau)$, so establishing that $\sigma \preceq_* \tau$. ($\subseteq$) To prove the inclusion $\preceq_* \subseteq \preceq_c \cap \preceq_s$, let us consider the triple $(\preceq_*, \preceq_*, \preceq_*)$. It is not difficult to check, using Theorem 3.9, that it is a coinductive client/server/peer triple, so that $\preceq_* \subseteq \preceq_c$ and $\preceq_* \subseteq \preceq_s$ by Theorem 3.9 again.

We end this section by defining a relation that is a model of the Gay-Hole subtyping relation on session types, as we prove in the next section.

Definition 3.11 (Semantic Subtyping)

i) We define $\mathcal{SB}_{|*}$ as the set $\mathcal{SB}$ restricted to session behaviours containing only the polarity '$*$';

ii) The semantic subtyping *relation $\preceq$: $\subseteq \mathcal{SB}_{|*} \times \mathcal{SB}_{|*}$ is defined as the relation $\preceq_*$ restricted to pairs in $\mathcal{SB}_{|*} \times \mathcal{SB}_{|*}$.*

Notice that $\preceq$: is in turn the restriction to $\mathcal{SB}_{|*}$ of the subsieve relation in [8].

4 CSP-Session Types and CSP-Subtyping.

In this section we extend the usual formalism of session types with delegation and their corresponding subtyping relation [16] in order to take into account the particular roles (client server or peer) played by a component during an interaction along a channel. We define a set of session type expressions which we call CSP-Session Types (Client/Server/Peer Session Types) and devise a formal system for deriving three kinds of inequalities to be interpreted as particular forms of subtyping that take in account the roles played by the users of channels. More precisely, we consider inequalities of the shapes $A \leq_c B$, $A \leq_s B$ and

$A \leq_* B$ for *client*, *server* and *peer subtyping*, respectively. In the next section we shall then provide a semantics of session types by means of behaviours in $\mathcal{SB}$, and show that the system is sound and complete w.r.t. the sub-behaviour relations $\preceq_c$, $\preceq_s$ and $\preceq_*$ respectively.

We shall consider session types only (live channel types), disregarding sorts in the terminology of [17].

Definition 4.1 (CSP-Session Types) *i) The set of* raw client/server/peer type expressions *is defined according to the following grammar:*

$$
\begin{array}{rcll}
p, q & ::= & c \mid s \mid * & \textit{polarities} \\
A, B & & & \textit{session types} \\
& ::= & \mathsf{end} & \textit{terminated session} \\
& \mid & \&\langle \ell_i : B_i \mid i \in I \rangle & \textit{branching} \\
& \mid & \oplus\langle \ell_i : B_i \mid i \in I \rangle & \textit{selection} \\
& \mid & ?(A^p)B & \textit{input} \\
& \mid & ![A^p]B & \textit{output} \\
& \mid & X & \textit{variable} \\
& \mid & \mu X.A & \textit{recursion}
\end{array}
$$

where

- *I is a finite and non empty set of indexes;*
- *the ℓ_i's belong to a denumerable set of* labels*;*
- *X is a session type variable out of a denumerable set, and it is bound in $\mu X.A$, free otherwise:* $\textsc{fv}(A)$ *is the set of free variables occurring in A;*
- *$p \in \{c, s, *\}$.*

ii) The set $\mathcal{ST}$ of Client/Server/Peer session types*,* CSP-types *for short, is defined as the set of of raw type expressions such that A is not a variable in $\mu X.A$; further $\textsc{fv}(A) = \emptyset$ in both $?(A^p)B$ and $![A^p]B$ expressions.*

iii) We define $\mathcal{ST}_{|*}$ as the set of CSP-types containing only the polarity '$*$'.

In definitions and in the technical treatment we consider only pure session types without ground types $G = \mathbf{Bool}, \mathbf{Int}, \ldots$; however ground types can be easily added to the syntax of input/output types by admitting the types $?(G^p)B$ and $?(G^p)B$.

In the following, $\&_{i \in I}\langle \ell_i : B_i \rangle$ and $\oplus_{i \in I}\langle \ell_i : B_i \rangle$ will be sometimes used as shorthand for, respectively, $\&\langle \ell_i : B_i \mid i \in I \rangle$ and $\oplus\langle \ell_i : B_i \mid i \in I \rangle$.

The restriction that A is not a variable in $\mu X.A$ is the usual one to make recursive types contractive. Beside this, CSP-session types extend ordinary session types syntax because of polarities in the type semantics, while they have been used only for live variables in [16, 23] as well as in our [2].

On the other hand we are more restrictive w.r.t. input/output types. The restriction that A must be closed in the contexts $?(A^p)B$ and $![A^p]B$ is new, and it is clearly connected with the behavioural semantics we are proposing in this paper. It rules out types e.g. of the shape $\mu X.?(X^p)A$ and $\mu X.![X^p]A$ which (by disregarding the polarity p) are legal session types in the literature. Since we adopt the equi-recursive approach in the treatment of recursive types (see [15]), if $X \notin \text{FV}(A)$ then the input (output respectively) of the type itself is immaterial for the continuation of type A, and we do not see any sensible use of such an input (or output). If instead $X \in \text{FV}(A)$ then setting $B = \mu X.?(X^p)A$ we have that B is isomorphic to $?(B^p)A\{B/X\}$ which, in turn, should be isomorphic to $\mu X.?(B^p)A$, being the latter a legal type in our context but for the B subexpression. Of course this is not a definite argument to exclude any theoretical loss because of the restriction we have adopted.

Definition 4.2 (CSP-Subtyping) *The* client subtyping*, denoted by $\leq_c$, the* server subtyping*, denoted by $\leq_s$, and the* peer subtyping*, denoted by $\leq_*$ are the binary relations on* closed *CSP-session types defined by the following rules, where $p, q \in \{c, s, *\}$.*
The symbol Γ in a judgment $\Gamma \vdash A \leq_p B$ denotes a finite set of type inequalities of the form $C \leq_p D$. As usual $\Gamma, C \leq_p D$ abbreviates $\Gamma \cup \{C \leq_p D\}$, assuming that $C \leq_p D \notin \Gamma$.

$$\Gamma \vdash A \leq_c \mathsf{end} \quad \text{(T-Ax-C)} \qquad \Gamma \vdash \mathsf{end} \leq_s A \quad \text{(T-Ax-S)}$$

$$\Gamma \vdash \mathsf{end} \leq_* \mathsf{end} \quad \text{(T-Ax-P)}$$

$$\Gamma \vdash A \leq_p A \ \text{(T-Sub-Id)} \qquad \Gamma, A \leq_p B \vdash A \leq_p B \ \text{(T-Sub-Hyp)}$$

$$\frac{\Gamma \vdash A\{\mu X.A/X\} \leq_p B}{\Gamma \vdash \mu X.A \leq_p B}\ \text{(T-Sub-Unf-L)} \qquad \frac{\Gamma \vdash B \leq_p A\{\mu X.A/X\}}{\Gamma \vdash B \leq_p \mu X.A}\ \text{(T-Sub-Unf-R)}$$

$$\frac{\Gamma, \&_{i\in I}\langle \ell_i : A_i\rangle \leq_p \&_{j\in J}\langle \ell_j : B_j\rangle \vdash A_i \leq_p B_i \quad \forall i \in I \quad I \subseteq J}{\Gamma \vdash \&_{i\in I}\langle \ell_i : A_i\rangle \leq_p \&_{j\in J}\langle \ell_j : B_j\rangle}\ \text{(T-Sub-\&)}$$

$$\frac{\Gamma, \oplus_{i\in I}\langle \ell_i : A_i\rangle \leq_p \oplus_{j\in J}\langle \ell_j : B_j\rangle \vdash A_j \leq_p B_j \quad \forall j \in J \quad I \supseteq J}{\Gamma \vdash \oplus_{i\in I}\langle \ell_i : A_i\rangle \leq_p \oplus_{j\in J}\langle \ell_j : B_j\rangle}\ \text{(T-Sub-}\oplus\text{)}$$

$$\frac{\Gamma,\ ?(A^q)B \leq_p ?(C^q)D \vdash A \leq_q C, B \leq_p D}{\Gamma \vdash ?(A^q)B \leq_p ?(C^q)D}\ \text{(T-Sub-In)}$$

$$\frac{\Gamma,\ ![A^q]B \leq_p ![C^q]D \vdash C \leq_q A, B \leq_p D}{\Gamma \vdash ![A^q]B \leq_p ![C^q]D}\ \text{(T-Sub-Out)}$$

We say that A is a client (server, peer) subtype *of B, written $A \leq_{c(s,*)} B$ whenever $\emptyset \vdash A \leq_{c(s,*)} B$ is derivable.*

We stress that the system above allows closed session types only. In particular we do not have to care about assumptions like $X \leq_p Y$ in the treatment of recursion, which was the case e.g. in Abadi and Cardelli's original subtyping system in [1].

The rules of our system are inspired to the axiomatisation of recursive simple types in [5], in that for each type constructor we have a rule that coinductively defines its meaning, whereas the μ-types are simply unfolded on the right-hand-side of the judgments. This has the technical advantage of being closer to the coinductive characterisation of the sub-behaviour relations studied in this paper.

Remark 4.3 *The system in Definition 4.2 is algorithmic because it satisfies a kind of subformula property (see Lemma 5.14 below for a precise statement), so that the upward reconstruction of the derivation can be syntax driven. It is also deterministic, but possibly for the choice of the order in which* (T-Sub-Unf-L) *and* (T-Sub-Unf-R) *are used as in the example 4.7, which is immaterial.*

4.1 Relationship between CSP and Gay-Hole Subtyping.

In the present subsection we show the relationship between our system and the Gay and Hole algorithmic subtyping system in [16], to which we refer for its definition. By what we previously discussed, we consider closed session types only also for the Gay-Hole system. We denote by $\mathcal{GHT}$ the usual set of session types with delegation used in [16], and by $\leq$ the algorithmic subtyping relation defined there. Then we claim that the sets $\mathcal{ST}_{|*}$ and $\mathcal{GHT}$ are essentially the same (see also below 5.3).

Definition 4.4 *We define* $\leq\!:$ *as the restriction of* $\leq_*$ *to* $\mathcal{ST}_{|*} \times \mathcal{ST}_{|*}$

In order to avoid too cumbersome a notation, we shall identify elements of $\mathcal{ST}_{|*}$ with the corresponding ones in $\mathcal{GHT}$.

Proposition 4.5 *Let* $\leq_*^-$ *be the relation defined on* $\mathcal{ST}_{|*}$ *by the formal system of Definition 4.2 without axioms* (T-Ax-C) *and* (T-Ax-S)*. Then, for* $A, B \in \mathcal{ST}_{|*}$*, we have*

i) $\quad \vdash A \leq B \quad \Leftrightarrow \quad \vdash A \leq_*^- B$

ii) $\quad \vdash A \leq_*^- B \quad \Leftrightarrow \quad \vdash A \leq\!: B$

The precise relationship between CSP-subtyping and the Gay-Hole algorithmic subtyping of [16] is stated then in the following corollary.

Corollary 4.6 *Let* $A, B \in \mathcal{ST}_{|*}$.

$$\vdash A \leq B \quad \Leftrightarrow \quad \vdash A \leq\!: B$$

Proposition 4.5(ii) easily descends from Remark 4.3. For what concerns Proposition 4.5(i), we do not prove it in detail, since it is rather routine and since the only difference between our system and the algorithmic one of [16] lies in the coinductive treatment of type constructors rules. We illustrate instead such different treatment by means of an example.

Example 4.7 *Given* $A = \mu X.\&\langle \ell_1 : X\rangle$ *and* $B = \mu X.\&\langle \ell_1 : X, \ell_2 : C\rangle$*, where* C *is any type in* $\mathcal{ST}_{\natural *}$*, we can derive:* $\vdash A \leq_* B$ *as follows:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\&\langle \ell_1 : A\rangle \leq_* \&\langle \ell_1 : B, \ell_2 : C\rangle \vdash \&\langle \ell_1 : A\rangle \leq_* \&\langle \ell_1 : B, \ell_2 : C\rangle}\,(\text{T-Sub-Hyp})}{\&\langle \ell_1 : A\rangle \leq_* \&\langle \ell_1 : B, \ell_2 : C\rangle \vdash \&\langle \ell_1 : A\rangle \leq_p B}\,(\text{T-Sub-Unf-R})}{\&\langle \ell_1 : A\rangle \leq_* \&\langle \ell_1 : B, \ell_2 : C\rangle \vdash A \leq_* B}\,(\text{T-Sub-Unf-L})}{\vdash \&\langle \ell_1 : A\rangle \leq_* \&\langle \ell_1 : B, \ell_2 : C\rangle}\,(\text{T-Sub-\&})}{\vdash \&\langle \ell_1 : A\rangle \leq_* B}\,(\text{T-Sub-Unf-R})}{\vdash A \leq_* B}\,(\text{T-Sub-Unf-L})$$

The same conclusion (after erasing the polarities) is derived in the system in [16] (Figure 11), using the rules:

$$\dfrac{\Gamma, \mu X.A \leq B \vdash A\{\mu X.A/X\} \leq B}{\Gamma \vdash \mu X.A \leq B}\,(\text{AS-Rec-L})$$

and its symmetric (AS-Rec-R) *to unfold the* B *above, plus*

$$\dfrac{\Gamma \vdash A_i \leq_p B_i \quad \forall i \in I \quad I \subseteq J}{\Gamma \vdash \&_{i\in I}\langle \ell_i : A_i\rangle \leq_p \&_{j\in J}\langle \ell_j : B_j\rangle}\,(\text{AS-Branch})$$

Then we can build the derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \leq B, \&\langle \ell_1 : A\rangle \leq B \vdash A \leq B}\,(\text{AS-Assump})}{A \leq B, \&\langle \ell_1 : A\rangle \leq B \vdash \&\langle \ell_1 : A\rangle \leq \&\langle \ell_1 : B, \ell_2 : C\rangle}\,(\text{AS-Branch})}{A \leq B \vdash \&\langle \ell_1 : A\rangle \leq B}\,(\text{AS-Rec-R})}{\vdash A \leq B}\,(\text{AS-Rec-L})$$

where (AS-Assump) *is the same axiom as* (T-Sub-Hyp)*. Also the complexity of the two derivations is comparable: the first derivation is higher than the second one in that it makes the unfoldings twice; on the other hand the second derivation has a larger set of assumptions where the unfoldings are saved, so that, looking at the systems algorithmically, space in the second system compensate time in the first one.*

5 Behavioural Semantics of CSP-Subtyping.

We devote now the present section to provide a behavioural semantics for CSP-session types and for CSP-subtyping. At the end we show how to use such semantics, togheter with the results of the previous subsection, in order to get also a semantics for Gay-Hole subtyping.

Given the definition of CSP-session types, it is straightforward to interpret them into behaviours.

Definition 5.1 (Behavioural Semantics of CSP-Session Types) *Assume, without loss of generality, that there is a bijective mapping $[\![X]\!] = x$ from type variables to behaviour variables, and that the set $\mathcal{N}$ is in one-to-one correspondence with the set of labels ℓ_i's used in session types.*
Then the semantic function $[\![\cdot]\!] : \mathcal{ST} \to \mathcal{SB}$ is obtained by restricting to session types the mapping from raw type expressions into raw behaviour expressions defined as follows:

$$\begin{array}{rcl}
[\![\mathsf{end}]\!] & = & \mathbf{1} \\
[\![\&\langle \ell_i : B_i \mid i \in I\rangle]\!] & = & \sum_{i\in I} \ell_i.[\![B_i]\!] \\
[\![\oplus\langle \ell_i : B_i \mid i \in I\rangle]\!] & = & \bigoplus_{i\in I} \overline{\ell}_i.[\![B_i]\!] \\
[\![X]\!] & = & x \\
[\![\mu X.\ A]\!] & = & \mathsf{rec}\,[\![X]\!].\,[\![A]\!] \\
[\![\,?(A^p)B]\!] & = & ?([\![A]\!]^p)[\![B]\!] \\
[\![\,!{[A^p]}B]\!] & = & ![[\![A]\!]^p][\![B]\!]
\end{array}$$

Ground types could be interpreted into $\mathcal{SB}$ by means of names in $\mathcal{N}$, so that we could add the clause $[\![G]\!] = G.\mathbf{1}$ to the definition of the semantic mapping. The subtyping theory is intended to formalize structural subtyping, so that no axiom like e.g. **Int** $\leq_p$ **Real** is ever considered. For practical purposes, however, it shouldn't be problematic to add axioms like **Int** $\leq_p$ **Real** without loosing the theoretical properties of the system established in this paper.

The interpretation mapping $[\![\cdot]\!]$ is well defined and basic syntactical properties are preserved in the following sense:

Lemma 5.2 *If $A \in \mathcal{ST}$ then $[\![A]\!] \in \mathcal{SB}$. In particular $\mathrm{FV}([\![A]\!]) = \{x \mid \exists X \in \mathrm{FV}(A).\ x = [\![X]\!]\}$, so that if $\mathrm{FV}(A) = \emptyset$ then $\mathrm{FV}([\![A]\!]) = \emptyset$; moreover, the mapping $[\![\cdot]\!]$ is well behaved w.r.t. substitution, that is $[\![A\{B/X\}]\!] = [\![A]\!]\{[\![B]\!]/[\![X]\!]\}$.*

Proof. By straightforward induction over the structure of A. □

It is immediate to check that we actually have the following

Fact 5.3 *The mapping $[\![\cdot]\!] : \mathcal{ST} \to \mathcal{SB}$ is a bijection.*

Remark 5.4 *When typing π-calculus terms, dual session types play an essential role (see [17, 23]). Roughly, the dual of a type is obtained by exchanging $\&$ with $\oplus$ and input with output. More precisely, if A is a type expression then define the* dual of A *as the type expression $\overline{A}$ by:*

$$\begin{array}{rcl}
\overline{\mathsf{end}} & = & \mathsf{end} \\
\overline{\&\langle \ell_i : B_i \mid i \in I\rangle} & = & \oplus\langle \ell_i : \overline{B}_i \mid i \in I\rangle \\
\overline{\oplus\langle \ell_i : B_i \mid i \in I\rangle} & = & \&\langle \ell_i : \overline{B}_i \mid i \in I\rangle \\
\overline{X} & = & X \\
\overline{\mu X.A} & = & \mu X.\overline{A} \\
\overline{?(A^p)B} & = & ![A^p]\overline{B} \\
\overline{![A^p]B} & = & ?(A^p)\overline{B}
\end{array}$$

Now, if $A \in \mathcal{ST}$ then $\overline{A} \in \mathcal{ST}$, moreover $[\![\overline{A}]\!] = \overline{[\![A]\!]}$, which is immediate by induction over the structure of A.

Given the interpretation of types, we proceed by defining the semantics of $\leq_c$, $\leq_s$ and $\leq_*$, and in general of judgments $\Gamma \vdash A \leq_p B$; recall that these are about closed types, both in Γ and in $A \leq_p B$:

Definition 5.5 (Judgment Semantics) *Let $p, q \in \{c, s, *\}$.*

i) $\models A \leq_p B$ *iff* $[\![A]\!] \preceq_p [\![B]\!]$

ii) $\models \Gamma$ *iff* $\models C \leq_q D$ *for all* $C \leq_q D \in \Gamma$

iii) $\Gamma \models A \leq_p B$ *iff* $\models \Gamma$ *implies* $\models A \leq_p B$

To facilitate the proofs below it is convenient to consider the following stratified version of Definition 5.5:

a) $\models_k A \leq_p B$ iff $[\![A]\!] \preceq_p^{co.k} [\![B]\!]$;

b) $\models_k \Gamma$ iff $\models_k C \leq_p D$ for all $C \leq_q D \in \Gamma$;

c) $\Gamma \models_k A \leq_p B$ iff $\models_k \Gamma$ implies $\models_k A \leq_p B$.

Note that the stratification is done w.r.t. the characterisation of the relations $\preceq_p$ in terms of the operator $\mathcal{H}$ in Theorem 3.9. In fact, by this theorem we have that $\models A \leq_p B$ if and only if $\models_k A \leq_p B$ for all k and hence:

$$\forall k. \Gamma \models_k A \leq_p B \;\Rightarrow\; \Gamma \models A \leq_p B.$$

This observation will be used in the proof of the following Soundness Theorem. The opposite implication does not hold. In fact once it is the case that $\not\models_k C \leq_p D$ for some C, D and some k, then the same is true for any $h \geq k$, but $\models_{h'} C \leq_p D$ for some $h' < k$, for $h' = 0$ at least. Therefore it is possible to choose Γ such that $\models_k \Gamma$ and $\not\models_k A \leq_p B$, while it is true that $\Gamma \models A \leq_p B$ just because $\not\models \Gamma$, which only means that $\not\models_h \Gamma$ for all but finitely many h. In the Completeness Theorem, however, only the fact that $\models A \leq_p B$ if and only if $\models_k A \leq_p B$ for all k is needed.

As usual we write ambiguously $\Gamma \vdash A \leq_p B$ for the judgment itself and the statement that it is derivable.

Theorem 5.6 (Soundness) *For any judgment $\Gamma \vdash A \leq_p B$, with $p = c, s, *$:*

$$\Gamma \vdash A \leq_p B \quad \Rightarrow \quad \Gamma \models A \leq_p B.$$

Proof. We actually prove $\Gamma \vdash A \leq_p B \Rightarrow \forall k.\ \Gamma \models_k A \leq_p B$ by a principal induction over the derivation of $\Gamma \vdash A \leq_p B$, and a subordinate induction over k.

In case of axioms (T-Ax-C), (T-Ax-S) or (T-Ax-P) the thesis holds since $[\![\mathsf{end}]\!] = \mathbf{1}$ and for all $\sigma = [\![A]\!]$ and all k we have $\sigma \preceq_c^{co.k} \mathbf{1}$, $\mathbf{1} \preceq_s^{co.k} \sigma$ and $\mathbf{1} \preceq_*^{co.k} \mathbf{1}$by definition of $\mathcal{H}$.

The cases of axioms (T-Sub-Id) and (T-Sub-Hyp) are obvious.

In case of rule (T-Sub-Unf-L) we observe that, if $[\![\mu X.A]\!] = \mathsf{rec}\, x.\sigma$ then $\mathsf{rec}\, x.\sigma \Downarrow \tau$ if and only if $\sigma\{\mathsf{rec}\, x.\sigma/x\} \Downarrow \tau$ hence $[\![\mu X.A]\!] \preceq_p^{co.k} [\![B]\!]$ if and only if $\sigma\{\mathsf{rec}\, x.\sigma/x\} = [\![A\{\mu X.A/X\}]\!] \preceq_p^{co.k} [\![B]\!]$, which holds by induction since it is the right-hand statement of the judgment in the hypothesis of the rule. The case of rule (T-Sub-Unf-R) is analogous.

Of the remaining cases T-Sub-&, T-Sub-In, T-Sub-$\oplus$ and T-Sub-Out we treat explicitly the first two, as the others are symmetric.

(T-Sub-&). $\Gamma \models_0 \&_{i\in I}\langle \ell_i : A_i\rangle \leq_p \&_{j\in J}\langle \ell_j : B_j\rangle$ is trivially true, since $[\![\&_{i\in I}\langle \ell_i : A_i\rangle]\!] \preceq_p^0 [\![\&_{j\in J}\langle \ell_j : B_j\rangle]\!]$ holds always.

To establish $\Gamma \models_{k+1} \&_{i\in I}\langle \ell_i : A_i\rangle \leq_p \&_{j\in J}\langle \ell_j : B_j\rangle$, assume $\models_{k+1} \Gamma$. Since $\preceq_p^k \supseteq \preceq_p^{k+1}$, we have that $\models_k \Gamma$. By the secondary induction hypothesis: $\Gamma \models_k \&_{i\in I}\langle \ell_i : A_i\rangle \leq_p \&_{j\in J}\langle \ell_j : B_j\rangle$, so that it must be the case that $\models_k \&_{i\in I}\langle \ell_i : A_i\rangle \leq_p \&_{j\in J}\langle \ell_j : B_j\rangle$.
It follows that $\models_k \Gamma, \&_{i\in I}\langle \ell_i : A_i\rangle \leq_p \&_{j\in J}\langle \ell_j : B_j\rangle$; hence by the primary induction hypothesis: $\Gamma, \&_{i\in I}\langle \ell_i : A_i\rangle \leq_p \&_{j\in J}\langle \ell_j : B_j\rangle \models_k A_i \leq_p B_i$ for all $i \in I$. We conclude that $\models_k A_i \leq_p B_i$ for all $i \in I$.

Let $[\![\&_{i\in I}\langle \ell_i : A_i\rangle]\!] = \sum_{i\in I} \ell_i.\sigma_i$, where $\sigma_i = [\![A_i]\!]$, and similarly $[\![\&_{j\in J}\langle \ell_j : B_j\rangle]\!] = \sum_{j\in J} \ell_j.\tau_j$ with $\tau_j = [\![B_j]\!]$. Then trivially $\sum_{i\in I} \ell_i.\sigma_i \Downarrow \sum_{i\in I} \ell_i.\sigma_i$ and $\sum_{j\in J} \ell_j.\tau_j \Downarrow \sum_{j\in J} \ell_j.\tau_j$. On the other hand we know that $\sigma_i \preceq_p^{co.k} \tau_i$ for all $i \in I$, since $\models_k A_i \leq_p B_i$. Hence $\sum_{i\in I} \ell_i.\sigma_i \preceq_p^{co.k+1} \sum_{j\in J} \ell_j.\tau_j$, that is $\models_{k+1} \&_{i\in I}\langle \ell_i : A_i\rangle \leq_p \&_{j\in J}\langle \ell_j : B_j\rangle$ as desired.

(T-Sub-In). $\Gamma \models_0 ?(A^p)B \leq_q ?(C^p)D$ since $[\![\, ?(A^p)B]\!] \preceq_q^{co.0} [\![\, ?(C^p)D]\!]$ holds always.

To establish $\Gamma \models_{k+1} ?(A^p)B \leq_q ?(C^p)D$, let us assume $\models_{k+1} \Gamma$.
Since $\preceq_p^{co.k} \supseteq \preceq_p^{co.k+1}$, we have that $\models_k \Gamma$. By the secondary induction hypothesis: $\Gamma \models_k ?(A^p)B \leq_q ?(C^p)D$, so that it must be the case that $\models_k ?(A^p)B \leq_q ?(C^p)D$. It follows that $\models_k \Gamma, ?(A^p)B \leq_q ?(C^p)D$; hence by the primary induction hypothesis both the statement $\Gamma, ?(A^p)B \leq_q ?(C^p)D \models_k B \leq_q D$ and $\Gamma, ?(A^p)B \leq_q ?(C^p)D \models_k A \leq_p C$ hold. We conclude that $\models_k B \leq_q D$ and $\models_k A \leq_p C$. Let $[\![\, ?(A^p)B]\!] = ?(\sigma_1)\sigma_2$ and $[\![\, ?(C^p)D]\!] = ?(\tau_1)\tau_2$, where $[\![A]\!] = \sigma_1$, $[\![B]\!] = \sigma_2$, $[\![C]\!] = \tau_1$ and $[\![D]\!] = \tau_2$. We trivially have that $?(A^p)B \Downarrow ?(A^p)B$ and $?(C^p)D \Downarrow ?(C^p)D$. On the other hand we know that $\sigma_1 \preceq_p^{co.k} \tau_1$ and $\sigma_2 \preceq_q^{co.k} \tau_2$, since $\models_k A \leq_p C$ and $\models_k B \leq_q D$. Hence $?(\sigma_1^p)\sigma_2 \preceq_q^{co.k+1} ?(\tau_1^p)\tau_2$, that is $\models_{k+1} ?(A^p)B \leq_q ?(C^p)D$, as desired.

Corollary 5.7 *For all $A, B \in \mathcal{ST}$ and $p = c, s, *$*

$$\vdash A \leq_p B \quad \Rightarrow \quad [\![A]\!] \preceq_p [\![B]\!].$$

Proof. By Theorem 5.6, taking $\Gamma = \emptyset$ and recalling that $\models A \leq_p B$ is defined as $[\![A]\!] \preceq_p [\![B]\!]$.

As far as completeness is concerned we preliminary observe that the inverse implication of Theorem 5.6 doesn't hold, namely $\Gamma \models A \leq_p B \not\Rightarrow \Gamma \vdash A \leq_p B$

in general. This is due to the fact that $\Gamma \models A \leq_p B$ is a conditional statement, which is vacuously true when $\not\models \Gamma$. This is the case if Γ contains some inequality $C \leq_q D$ such that $\not\models C \leq_q D$, that is $[\![C]\!] \not\leq_q [\![D]\!]$. Now consider the inequalities $\mathsf{end} \leq_c \&\langle \ell : \mathsf{end}\rangle$ and $\&\langle \ell : \mathsf{end}\rangle \leq_s \mathsf{end}$: then it is easy to see that $\not\models \mathsf{end} \leq_c \&\langle \ell : \mathsf{end}\rangle$ (and by the way also $\not\models \&\langle \ell : \mathsf{end}\rangle \leq_s \mathsf{end}$); but $\mathsf{end} \leq_c \&\langle \ell : \mathsf{end}\rangle \models \&\langle \ell : \mathsf{end}\rangle \leq_s \mathsf{end}$ vacuously, whereas $\mathsf{end} \leq_c \&\langle \ell : \mathsf{end}\rangle \not\vdash \&\langle \ell : \mathsf{end}\rangle \leq_s \mathsf{end}$ because it is neither an instance of an axiom nor of the conclusion of any rule.

Nonetheless we are able to prove a completeness theorem, namely the statement that $\models A \leq_p B$ implies $\vdash A \leq_p B$, which is done below. We begin by introducing some auxiliary concepts.

Definition 5.8 *Given a finite set Γ of (closed) type inequalities and a (closed) type inequality $A \leq_p B$ with $p = c, s, *$, we define, for any $k \in \mathbb{N}$, the predicate $\Gamma \vdash_k A \leq_p B$ by:*

i) $\Gamma \vdash_k A \leq_p B$ *holds if either* $\Gamma \vdash A \leq_p B$ *is an axiom, or* $k = 0$*;*

ii) $\Gamma \vdash_k A \leq_p B$ *holds if there is an instance of either* (T-Sub-Unf-L) *or* (T-Sub-Unf-R) *inference rules*

$$\frac{\Gamma \vdash A' \leq_p B'}{\Gamma \vdash A \leq_p B}$$

such that $\Gamma \vdash_k A' \leq_p B'$*;*

iii) $\Gamma \vdash_{k+1} A \leq_p B$ *holds if there is an instance of any inference rule*

$$\frac{\Gamma_1 \vdash A_1 \leq_p B_1 \quad \cdots \quad \Gamma_n \vdash A_n \leq_p B_n}{\Gamma \vdash A \leq_p B}$$

but of rules (T-Sub-Unf-L) *and* (T-Sub-Unf-R)*, such that* $\Gamma_i \vdash_k A_i \leq_p B_i$ *for all* $i \in \{1, \ldots, n\}$*.*

The following remark helps to understand the meaning of the judgments $\Gamma \vdash_k A \leq_p B$.

Remark 5.9 *Let us consider derivation trees whose nodes are labeled by subtyping judgments and such that each internal node represents the conclusion of a rule and its immediate descendants its premises. Hence a derivation is a finite derivation tree whose leaves are (instances of) axioms. Then the meaning of $\Gamma \vdash_k A \leq_p B$ is that there exists a finite derivation tree $\mathcal{D}$ with conclusion $\Gamma \vdash A \leq_p B$ which possibly is not a derivation, as it could have some leafs which are not axioms. The index k is a bound to the number of the rules in a branch of $\mathcal{D}$ other than* (T-Sub-Unf-L) *and* (T-Sub-Unf-R)*. Of course if $\Gamma \vdash_k A \leq_p B$ then for any $h \leq k$ $\Gamma \vdash_h A \leq_p B$ holds, and its tree $\mathcal{D}'$ can be chosen so that $\mathcal{D}' \subseteq \mathcal{D}$, when considered as prefix closed sets of (labeled) nodes. In fact, because of Remark 4.3, the derivation system is syntax directed but when both types in the right-hand-side inequation are μ-types. Such types, however, must be unfolded in order to match the conclusion of any rule, and hence there can be ambiguity only about the order of rules* (T-Sub-Unf-L) *and* (T-Sub-Unf-R)*, which is immaterial.*

If $\Gamma \vdash A \leq_p B$ then $\Gamma \vdash_k A \leq_p B$ for any k, and in fact the derivation of $\Gamma \vdash A \leq_p B$ is exactly the derivation tree establishing $\Gamma \vdash_{k'} A \leq_p B$ for some suitably large k'. On the other hand it is not necessarily the case that the branches of the derivation tree proving $\Gamma \vdash_k A \leq_p B$ can be extended to reach a true derivation of $\Gamma \vdash A \leq_p B$; in the case branches can be extended, this cannot be done infinitely many times, as it will be shown at the end of the completeness proof.

The predicate $\Gamma \vdash_k A \leq_p B$ is well defined by induction over k: this follows by the fact that μ-types are contractive, hence condition (ii) of Definition 5.8 cannot be satisfied infinitely many times. Contractivity also implies that the following mapping over $\mathcal{ST}$ (also considered in [16]) is well defined:

$$\mathsf{unfold}(A) = \begin{cases} \mathsf{unfold}(B\{A/X\}) & \text{if } A = \mu X.B \\ A & \text{otherwise.} \end{cases}$$

Indeed by contractivity of μ-types we know that any such a type is of the form $\mu X_1 \ldots \mu X_n.A$ where A is neither a variable nor a μ-type. It follows that $\mathsf{unfold}(\mu X_1 \ldots \mu X_n.A)$ is always defined.

Lemma 5.10 *For any $k \in \mathbb{N}$*

$$\Gamma \vdash_k A \leq_p B \quad \Leftrightarrow \quad \Gamma \vdash_k \mathsf{unfold}(A) \leq_p \mathsf{unfold}(B).$$

Proof. The thesis is a consequence of Definition 5.8 by observing that, reading derivations in the upward sense, no μ-type can be ever used before unfolding it through occurrences of either rule (T-Sub-Unf-L) or (T-Sub-Unf-R), and that the index k does not decrease right in that cases.

Lemma 5.11 *For any $A \in \mathcal{ST}$:*

$$[\![A]\!] \Downarrow [\![\mathsf{unfold}(A)]\!].$$

Proof. This is an immediate consequence of the fact that $[\![\mu X.A]\!] = \mathsf{rec}\, x.[\![A]\!]$, where $x = [\![X]\!]$, and that the semantic interpretation is well behaved w.r.t. substitution by Lemma 5.2.

For the proof of Lemma 5.12 below, the following notation will be useful. Let $A \in \mathcal{ST}$ be neither a variable nor a μ-type. Then we write $\mathsf{op}(A)$ to denote the main type constructor of A; in particular we have $\mathsf{op}(\mathsf{end}) = \mathsf{end}$. If instead A is a μ-type then we put $\mathsf{op}(A) = \mathsf{op}(\mathsf{unfold}(A))$. By this convention and the definition of $\mathsf{unfold}(A)$ we have that $\mathsf{op}(A) = \mathsf{op}(\mathsf{unfold}(A))$ for any A which is not a variable, and for closed A a fortiori.

Lemma 5.12 *For any $k \in \mathbb{N}$:*

$$\models_k \Gamma, A \leq_p B \quad \Rightarrow \quad \Gamma \vdash_k A \leq_p B$$

Proof. In the following we assume that $A \leq_p B \notin \Gamma$, since otherwise $\Gamma \vdash_k A \leq_p B$ holds trivially by definition because of the axiom (T-Sub-Hyp).

We prove the thesis by induction over k. When $k = 0$ the thesis holds immediately since $\Gamma \vdash_0 A \leq_p B$ is always true. Let $k > 0$: by Lemma 5.11 $\Gamma \vdash_k A \leq_p B$ if and only if $\Gamma \vdash_k \mathsf{unfold}(A) \leq_p \mathsf{unfold}(B)$; then it suffices to establish the latter proceeding by cases of $\mathsf{op}(A)$ and $\mathsf{op}(B)$, which are the principal type constructors of $\mathsf{unfold}(A)$ and $\mathsf{unfold}(B)$ respectively.

$\mathsf{op}(A) = \mathsf{end}$ and $\mathsf{op}(B) \neq \mathsf{end}$, i.e. $\mathsf{unfold}(A) = \mathsf{end}$ and $\mathsf{unfold}(B) \neq \mathsf{end}$. If $p = s$ then $\Gamma \vdash \mathsf{end} \leq_s B$ is an instance of the (T-Ax-S) axiom hence $\Gamma \vdash_k \mathsf{end} \leq_s B$ holds immediately. If instead $p = c$ first observe that $[\![A]\!] \Downarrow [\![\mathsf{end}]\!]$ by Lemma 5.11, and $[\![\mathsf{end}]\!] = \mathbf{1}$. Now $\models_k \Gamma, A \leq_c B$ implies $\models_k A \leq_c B$ and hence $\models_k \mathsf{end} \leq_p B$ i.e. $\mathbf{1} \preceq_c^{co.k} [\![B]\!]$. By definition of $\preceq_p^{co.k}$ this is only possible if $[\![B]\!] \Downarrow \mathbf{1}$ so that $\mathsf{unfold}(B) = \mathsf{end}$ and $\Gamma \vdash_k \mathsf{end} \leq_p \mathsf{end}$ is an instance of the axiom (T-Sub-Id).

$\mathsf{op}(B) = \mathsf{end}$ and $\mathsf{op}(A) \neq \mathsf{end}$: this is similar to the previous case, but for using axiom (T-Ax-C) in place of (T-Ax-S).

$\mathsf{op}(A) = \mathsf{op}(B) = \mathsf{end}$. If $p = s, c$ then we proceed as in the previous cases. If $p = *$ then $\Gamma \vdash \mathsf{end} \leq_* \mathsf{end}$ is an instance of the (T-Ax-P) axiom and hence $\Gamma \vdash_k \mathsf{end} \leq_* \mathsf{end}$ holds immediately.

$\mathsf{op}(A) = \&$: then $\mathsf{unfold}(A) = \&\langle \ell_i : A_i \mid i \in I \rangle$ for certain A_i, so that $[\![A]\!] \Downarrow [\![\&\langle \ell_i : A_i \mid i \in I \rangle]\!]$ by Lemma 5.11, where $[\![\&\langle \ell_i : A_i \mid i \in I \rangle]\!] = \sum_{i \in I} \ell_i.[\![A_i]\!]$. By hypothesis we know that $\models_k \Gamma, A \leq_p B$, which implies by the above that $[\![\mathsf{unfold}(B)]\!] = [\![\&\langle \ell_j : B_j \mid j \in J \rangle]\!] = \sum_{j \in J} \ell_j.[\![B_j]\!]$ where $\sum_{i \in I} \ell_i.[\![A_i]\!] \preceq_p^{co.k} \sum_{j \in J} \ell_j.[\![B_j]\!]$. It follows that, by definition of $\preceq_p^{co.k}$, $I \subseteq J$ and $[\![A_i]\!] \preceq_p^{co.k-1} [\![B_i]\!]$ for all $i \in I$.

Since $\preceq_p^{co.k} \subseteq \preceq_p^{co.k-1}$, we have that $\models_k \Gamma, A \leq_p B$ implies $\models_{k-1} \Gamma, A \leq_p B$ and in particular $\models_{k-1} \&\langle \ell_i : A_i \mid i \in I \rangle \leq_p \&\langle \ell_j : B_j \mid j \in J \rangle$. By the above we conclude that $\models_{k-1} \Gamma, \&\langle \ell_i : A_i \mid i \in I \rangle \leq_p \&\langle \ell_j : B_j \mid j \in J \rangle, A_i \leq_p B_i$ for all $i \in I$. By induction this implies that

$$\Gamma, \&\langle \ell_i : A_i \mid i \in I \rangle \leq_p \&\langle \ell_j : B_j \mid j \in J \rangle \vdash_{k-1} A_i \leq_p B_i$$

for all $i \in I$; but since $I \subseteq J$ we conclude by rule (T-Sub-&) that

$$\Gamma \vdash_k \&\langle \ell_i : A_i \mid i \in I \rangle \leq_p \&\langle \ell_j : B_j \mid j \in J \rangle$$

as desired.

$\mathsf{op}(B) = \&$: this is symmetric to the previous case.

The remaining cases of $\mathsf{op}(A)$ and $\mathsf{op}(B)$ can be treated in the same manner.

In order to complete the proof of completeness, it remains to prove that if $\Gamma \vdash_k A \leq_p B$ for all k then $\Gamma \vdash A \leq_p B$. This is done with minor differences by means of the same proof as Lemma 10 in [16], which in turn adapts to session types the proof in [22], Lemma 2.4.1.

Definition 5.13 (Subterms) *For any (in general open) $A \in \mathcal{ST}$ define the set* $\mathsf{Sub}(A)$ *of* sub-expressions *of A by the following clauses:*

i) $\mathsf{Sub}(\mathsf{end}) = \{\mathsf{end}\}$,

ii) $\mathsf{Sub}(\&\langle \ell_i : A_i \mid i \in I\rangle) = \{\&\langle \ell_i : A_i \mid i \in I\rangle\} \cup \bigcup_{i\in I} \mathsf{Sub}(A_i)$,

iii) $\mathsf{Sub}(\oplus\langle \ell_i : A_i \mid i \in I\rangle) = \{\oplus\langle \ell_i : A_i \mid i \in I\rangle\} \cup \bigcup_{i\in I} \mathsf{Sub}(A_i)$,

iv) $\mathsf{Sub}(\,?(T^p)B) = \{\,?(T^p)B\} \cup \mathsf{Sub}(T) \cup \mathsf{Sub}(B)$,

v) $\mathsf{Sub}(\,![T^p]B) = \{\,![T^p]B\} \cup \mathsf{Sub}(T) \cup \mathsf{Sub}(B)$,

vi) $\mathsf{Sub}(\mu X.B) = \{C\{\mu X.B/X\} \mid C \in \mathsf{Sub}(B)\}$,

vii) $\mathsf{Sub}(X) = \{X\}$.

Given the judgments $\Gamma \vdash A \leq_p B$ and $\Gamma' \vdash C \leq_q D$ define

$$\Gamma \vdash A \leq_p B \ \lhd_R \ \Gamma' \vdash C \leq_q D$$

if and only if $\Gamma \vdash A \leq_p B$ is not an instance of any axiom and $\Gamma \vdash A \leq_p B$ and $\Gamma' \vdash C \leq_q D$ are instances of the conclusion and of a premise of rule (R) respectively. Further define $\Gamma \vdash A \leq_p B \lhd \Gamma' \vdash C \leq_q D$ if $\Gamma \vdash A \leq_p B \lhd_R \Gamma' \vdash C \leq_q D$ for some R.

In the following the following notations willbe used:

$$\begin{aligned} \mathsf{Sub}(A \leq_p B) &= \mathsf{Sub}(A) \cup \mathsf{Sub}(B) \\ \mathsf{Sub}(\Gamma) &= \bigcup\{\mathsf{Sub}(C \leq_p D) \mid C \leq_p D \in \Gamma\} \end{aligned}$$

Lemma 5.14 *Suppose that $\Gamma \vdash A \leq_p B \ \lhd_R \ \Gamma' \vdash C \leq_q D$, then:*

i) $\Gamma \subseteq \Gamma'$

ii) $\mathsf{Sub}(C \leq_q D) \subseteq \mathsf{Sub}(A \leq_p B)$,

iii) $\Gamma' \subseteq \mathsf{Sub}(\Gamma) \cup \mathsf{Sub}(A \leq_p B)$.

Proof. By direct inspection of the rules.

Lemma 5.15 *There exists no infinite chain of typing judgments w.r.t. $\lhd$.*

Proof. Toward a contradiction, let $\Gamma_0 \vdash A_0 \leq_{p_0} B_0 \lhd_{R_0} \Gamma_1 \vdash A_1 \leq_{p_1} B_1 \lhd_{R_1} \cdots$ be such an infinite chain. Because of the contractiveness of the μ-types, A_k, B_k cannot always be μ-types from a given index on, and hence for every i there exists $j \geq i$ such that $R_j \notin \{\text{T-Sub-Unf-L}, \text{T-Sub-Unf-R}\}$, and in particular there exist infinitely many such R_j's. Being this the case, for all such j's we have, by definition of $\mathsf{Sub}()$, that $A_j \in \mathsf{Sub}(A_j)$, $B_j \in \mathsf{Sub}(B_j)$. Moreover, we have necessarily that $A_j \leq_{p_j} B_j \in \Gamma_{j+1} \setminus \Gamma_j$, since otherwise $\Gamma_j \vdash A_j \leq_{p_j} B_j$ would be an instance of axiom (T-Sub-Hyp). From this fact, together with Lemma 5.14(i), it follows that the cardinalities $|\Gamma_i|$ are unbounded. However, by repeated applications of Lemma 5.14(iii), we know that, for all i, $\Gamma_i \subseteq \Gamma_0 \cup \mathsf{Sub}(A_0 \leq_{p_0} B_0)$ which is a finite set.

Corollary 5.16 *Let $p = c, s, *$. If $\Gamma \vdash_k A \leq_p B$ for all k then $\Gamma \vdash A \leq_p B$.*

Proof. If $\Gamma \vdash_k A \leq_p B$ then there exists a derivation tree $\mathcal{D}_k$ with conclusion $\Gamma \vdash A \leq_p B$ and possibly some leaf which is not an axiom. By Remark 5.9 we can choose these trees so that $\mathcal{D}_h \subseteq \mathcal{D}_k$ whenever $h \leq k$, where $\mathcal{D}_h$ and $\mathcal{D}_k$ are viewed as prefix closed sets of nodes. Now, let $\mathcal{D} = \bigcup_k \mathcal{D}_k$. If $\Gamma \not\vdash A \leq_p B$ then $\mathcal{D}$ is infinite, hence it has an infinite branch by König Lemma since it is a finitary tree. But then the judgments labeling such a branch would be an infinite chain w.r.t. $\lhd$, contradicting Lemma 5.15.

Theorem 5.17 (Completeness) *Let $\Gamma, A \leq_p B$ be a finite set of type inequalities among closed types, with $p = c, s, *$, then:*

$$\Gamma \models A \leq_p B \quad \Leftrightarrow \quad (\not\models \Gamma \ \vee \ \Gamma \vdash A \leq_p B \ ).$$

Proof. ($\Leftarrow$) If $\not\models \Gamma$ then $\Gamma \models A \leq_p B$ holds vacuously; if instead $\Gamma \vdash A \leq_p B$ then $\Gamma \models A \leq_p B$ follows by Theorem 5.6.

($\Rightarrow$) If $\Gamma \models A \leq_p B$ and $\models \Gamma$ then $\models \Gamma, A \leq_p B$, and hence $\models_k \Gamma, A \leq_p B$ for all k. By Lemma 5.12 this implies $\Gamma \vdash_k A \leq_p B$ for all k, and then $\Gamma \vdash A \leq_p B$ by Corollary 5.16.

We get now as Corollaries, a number of results.

Corollary 5.18

i) The relations $\preceq_s$, $\preceq_c$ and $\preceq_*$ over $\mathcal{SB}$ are decidable.

ii) For all closed $A, B \in \mathcal{ST}$ and $p = c, s, *$:

$$\vdash A \leq_p B \ \Leftrightarrow \ [\![A]\!] \preceq_p [\![B]\!].$$

iii)

$$\vdash A \leq_* B \quad \Leftrightarrow \quad \vdash A \leq_c B \ \& \vdash A \leq_s B$$

Proof. (i) By Point (ii) and Fact 5.3, since $\vdash A \leq_p B$ is decidable by Lemma 5.15.
(ii) By Theorem 5.17, taking $\Gamma = \emptyset$.
(iii) By Theorem 3.10 and point (ii).

We end our work by showing that the Gay and Hole subtyping relation is sound and complete w.r.t. to our semantic subtyping of Definition 3.11.

Recall that we identify elements of $\mathcal{ST}_{|*}$ and the corresponding ones in $\mathcal{GHT}$.

Corollary 5.19 (Soundness and Completeness of Gay and Hole subtyping) *Let $A, B \in \mathcal{ST}_{|*}$.*

$$\vdash A \leq B \quad \Leftrightarrow \quad [\![A]\!] \preceq: [\![B]\!]$$

Proof.

$$\begin{array}{lll}
\vdash A \leq B & \Leftrightarrow \vdash A \leq: B & \text{(Cor. 4.6)} \\
& \Leftrightarrow \vdash A \leq_* B & \text{(Def. 4.2)} \\
& \Leftrightarrow \vdash A \leq_c B \ \& \vdash A \leq_s B & \text{(Cor. 5.18(iii))} \\
& \Leftrightarrow [\![A]\!] \preceq_c [\![B]\!] \ \& \ [\![A]\!] \preceq_s [\![B]\!] & \text{(Cor. 5.18(ii))} \\
& \Leftrightarrow [\![A]\!] \preceq_* [\![B]\!] & \text{(Th. 3.10)} \\
& \Leftrightarrow [\![A]\!] \preceq: [\![B]\!] & \text{(Def. 3.11)}
\end{array}$$

Remark 5.20 *In [3] it was erroneously stated that Gay and Hole subtyping can be modeled by the relation $\preceq_c \cap \preceq_s$. Such a statement, as it is, does not hold. In fact it is enough to consider the following counterexample: let us consider the behaviours $?(D^c).\mathsf{end}$ and $?(\mathsf{end}^c).\mathsf{end}$, where D is an arbitrary behaviour different from* end*. In our typing system it is easy to show that both*

$$?(D^c).\mathsf{end} \leq_c ?(\mathsf{end}^c).\mathsf{end} \qquad \textit{and} \qquad ?(D^c).\mathsf{end} \leq_s ?(\mathsf{end}^c).\mathsf{end}$$

hold, whereas at the same time we have that

$$?(D).\mathsf{end} \not\leq ?(\mathsf{end}).\mathsf{end} \qquad \textit{and} \qquad ?(D^c).\mathsf{end} \not\leq ?(\mathsf{end}^c).\mathsf{end}$$

where $\leq$ is the Gay-Hole subtyping relation[1]*.*

Actually a model that can be obtained for Gay and Hole subtyping from the system without $\preceq_*$ is the relation $\preceq_c \cap \breve{\preceq}_s$, where $\breve{\preceq}_s$ is defined by $\sigma \breve{\preceq}_s \tau \Leftrightarrow \breve{\sigma} \preceq_s \breve{\tau}$, where $\breve{\sigma}$ is obtained from σ by exchanging all the polarities in σ (i.e. by replacing c by s and vice versa). Intuitively, the use of $\breve{(\cdot)}$ in the relation $\preceq_c \cap \breve{\preceq}_s$ forces the synchronizing higher-order actions to be the servers of each other.

6 Related Work

The starting point of the present work are the subtyping system for session types in [16] and the contract theory in [7, 18, 11, 10]. In particular the concept of compliance originates from [18]; however, we depart from the original definition adopting a more general concept, such that even non terminating behaviours can be compliant. In fact we allow the satisfaction in the limit of the requirement that all the actions by the client should find an adequate reply by the server. This is similar to [21], and we give a definition which is literally the same as that one used in [20].

On the other hand, the concept of "subsession" from [21] (which is the same as that of "compliance" in [10]) is not our compliance, nor one of our sub-behaviour relations. Rather it is comparable to our orthogonality and behavioural subtyping, since for the test to succeed it is required that both sides of a parallel combination complete (reach a final state).

With respect to session type subtyping as presented in [16] we have used a slightly different version of the algorithmic subtyping system, inspired to the

[1] Such a counterexample was pointed out to us by an anonymous referee of a draft version of the present paper, whom we thank.

subtyping system in [5] for the simple types with arrow and μ-types. The resulting system, thought more verbose than the original one, is closer to the coinductive characterisation of the sub-behaviour preorders and technically more suitable for our treatment. In the study of the proof theoretical properties of the system we have largely profited of [22] and [15].

The issue of comparing contracts to session types has been addressed in [19, 9], besides the quoted [21, 10]. The choice of restricting to session behaviours is responsible for the neat characterization of the main concepts involved, and first of all of the notion of the dual of a behaviour. To appreciate the advantage of the definition of session behaviours one could compare it to the difficult treatment of duality for the full set of behaviours in [19] and the fact the the encodings from session types to contracts and viceversa are not inverse each other. A restriction to contracts, producing an effect similar to the one induced by our restrictions, has been proposed in [6] where terms like $a + b.c$ and also like $a + \overline{b}$ are avoided by imposing any output action $\overline{b}$ to be preceded by an internal tau action; however, the absence of an internal choice and the ability of mixing input (i.e. branching actions in our interpretation) and output summands (which are naturally interpreted as selection actions) make this behaviour calculus rather unsuitable for our purposes.

Finally, the recent work [4] elaborates on the sub-behaviour preorders and on the definition of session behaviours considered in the present paper, as well as in the former [3], and shows that the intersection of server and client sub-behaviour is a fully abstract model of subtyping of first order session types when restricted to "session contracts", which coincide with first-order behaviours in $\mathcal{SB}$. This result (but not the proof) is the same as our completeness Theorem and Corollary 5.18(ii), which are however more general.

7 Conclusion

We have considered a behavioural semantics of session types interpreted as a suitable kind of contracts, with higher-order input/output of contracts with labels that express the role (client, server or peer) played by the user of a sent/received "component" exhibiting that contract. Moving from the concept of compliance from contract theory, it is possible to define the notions of client, server and peer (this last notion being the intersection of the first two). Three sub-behaviour preorders can then be defined as the inclusion of the sets of clients, servers and peers, respectively. We have shown that the three preorders provide a sound and complete semantics for an extension with roles of Gay and Hole's subtyping theory of session types. Such a system is decidable, so that the sub-behaviour relations we have studied are decidable as well. As a by-product, also the original theory of subtyping has a complete model by interpreting subtyping as a suitable restriction of the peer sub-behaviour relation, with the mild restriction that input/output types have to be closed.

References

[1] R. M. Amadio and L. Cardelli. Subtyping recursive types. *ACM Trans. Program. Lang. Syst.*, 15(4):575–631, 1993.

[2] F. Barbanera, S. Capecchi, and U. de'Liguoro. Typing asymmetric client-server interaction. In *FSEN*, volume 5961 of *LNCS*, pages 97–112. Springer, 2009.

[3] F. Barbanera and U. de'Liguoro. Two notions of sub-behaviour for session-based client/server systems. In *Proceedings of PPDP'10*, pages 155–164. ACM, 2010.

[4] G. Bernardi and M. Hennessy. Modelling session types using contracts. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 1941–1946, New York, NY, USA, 2012. ACM.

[5] M. Brandt and F. Henglein. Coinductive axiomatization of recursive type equality and subtyping. *Fundam. Inform.*, 33(4):309–338, 1998.

[6] M. Bravetti and G. Zavattaro. A foundational theory of contracts for multi-party service composition. *Fundam. Inform.*, 89(4):451–478, 2008.

[7] S. Carpineti, G. Castagna, C. Laneve, and L. Padovani. A formal account of contracts for Web Services. In *WS-FM, 3rd Int. Workshop on Web Services and Formal Methods*, number 4184 in LNCS, pages 148–162. Springer, 2006.

[8] G. Castagna, M. Dezani-Ciancaglini, E. Giachino, and L. Padovani. General Session Types. Available from: `http://www.sti.uniurb.it/padovani/publications.html`, 2008.

[9] G. Castagna, M. Dezani-Ciancaglini, E. Giachino, and L. Padovani. Foundations of session types. In *PPDP*, pages 219–230. ACM, 2009.

[10] G. Castagna, N. Gesbert, and L. Padovani. Contracts for mobile processes. In *Proceedings of the 20th International Conference on Concurrency Theory (CONCUR'09)*, volume 5710 of *LNCS*, pages 211–228. Springer, 2009.

[11] G. Castagna, N. Gesbert, and L. Padovani. A theory of contracts for web services. *ACM Trans. Program. Lang. Syst.*, 31(5):19:1–19:61, July 2009.

[12] R. De Nicola and M. Hennessy. Testing equivalence for processes. In *ICALP*, volume 154 of *LNCS*, pages 548–560. Springer, 1983.

[13] R. De Nicola and M. Hennessy. CCS without tau's. In *TAPSOFT, Vol.1*, volume 249 of *LNCS*. Springer, 1987.

[14] M. Dezani-Ciancaglini, U. de' Liguoro, and N. Yoshida. On Progress for Structured Communications. In G. Barthe and C. Fournet, editors, *TGC'07*, volume 4912 of *LNCS*, pages 257–275. Springer, 2008.

[15] V. Gapeyev, M. Y. Levin, and B. C. Pierce. Recursive subtyping revealed. *J. Funct. Program.*, 12(6):511–548, 2002.

[16] S. Gay and M. Hole. Subtyping for Session Types in the Pi-Calculus. *Acta Informatica*, 42(2/3):191–225, 2005.

[17] K. Honda, V. T. Vasconcelos, and M. Kubo. Language Primitives and Type Disciplines for Structured Communication-based Programming. In *ESOP'98*, volume 1381 of *LNCS*, pages 22–138. Springer-Verlag, 1998.

[18] C. Laneve and L. Padovani. The Must Preorder Revisited: An Algebraic Theory for Web Services Contracts. In *CONCUR'07*, volume 4703 of *LNCS*, pages 212–225. Springer-Verlag, 2007.

[19] C. Laneve and L. Padovani. The pairing of contracts and session types. In *Concurrency, Graphs and Models*, volume 5065 of *LNCS*, pages 681–700, 2008.

[20] L. Padovani. Contract-based discovery and adaptation of web services. In M. Bernardo, L. Padovani, and G. Zavattaro, editors, *SFM*, volume 5569 of *LNCS*, pages 213–260. Springer, 2009.

[21] L. Padovani. Session types at the mirror. In *ICE*, volume 12 of *EPTCS*, pages 71–86, 2009.

[22] B. C. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. In *Logic in Computer Science*, 1993. Full version in *Mathematical Structures in Computer Science* , Vol. 6, No. 5, 1996.

[23] N. Yoshida and V. T. Vasconcelos. Language Primitives and Type Disciplines for Structured Communication-based Programming Revisited. In *SecReT'06*, volume 171 of *ENTCS*, pages 73–93. Elsevier, 2007.